POLYPUBLIE
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Caractérisation et amélioration de la testabilité séquentielle pseudo-aléatoire des circuits VLSI
Auteur: Author:	Mohamed Soufi
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Soufi, M. (1997). Caractérisation et amélioration de la testabilité séquentielle pseudo-aléatoire des circuits VLSI [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6940/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6940/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

CARACTÉRISATION ET AMÉLIORATION DE LA TESTABILITÉ SÉQUENTIELLE PSEUDO-ALÉATOIRE DES CIRCUITS VLSI

MOHAMED SOUFI
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSPHIÆ DOCTOR (PH. D.)
(GÉNIE ÉLECTRIQUE)
JANVIER 1997

©Mohamed Soufi 1997

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-33030-3

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Cette thèse intitulée:

CARACTÉRISATION ET AMÉLIORATION DE LA TESTABILITÉ SÉQUENTIELLE PSEUDO-ALÉATOIRE DES CIRCUITS VLSI

présentée par: SOUFI Mohamed
en vue de l'obtention du diplôme de: Philosophiae Doctor (Ph.D.)
a été dûment acceptée par le jury d'examen constitué de:

M. BOIS Guy, Ph. D., président
M. SAVARIA Yvon, Ph. D., membre et directeur de recherche
Mme KAMINSKA Bozena, Ph. D., membre et co-directrice de recherche
M. SAMAN Adham, Ph. D., membre externe
M. SAWAN Mohamad, Ph.D., membre

Je dédie cette thèse à toute ma famille, en particulier à ma mère, à ma mère, à ma mère et à mon défunt père (R. A.).

REMERCIEMENTS

Nous tenons à remercier ceux et celles qui, de près ou de loin, ont contribué à ce travail, en particulier: M. Yvon Savaria, dont la direction a été fondamentale dans l'élaboration de cette thèse. Je tiens surtout à le remercier pour les réunions extraordinaires à partir de 10h du soir. Je tiens également à remercier Mme. Bozena Kaminska d'avoir accepté de co-diriger la présente thèse, le Département de Génie Électrique et de Génie Informatique de l'École Polytechnique de Montréal ainsi que les membres de jury qui ont accepté d'évaluer le contenu de cette thèse.

Mes remerciements vont également à ma femme pour son support tout au long de cette thèse et pour avoir accepté le supplice de corriger mon horrible "latin". Mes remerciements vont de même à l'ensemble de ma famille pour leur soutien, leurs encouragements et leurs prières tout au long de mes études; ainsi qu'à tous ceux qui ont eu de la gentillesse, aux moments critiques, de céder une station, répondre à une question, etc...

RÉSUMÉ

Cette thèse a comme objectif l'étude, la caractérisation et le développement de nouvelles méthodes d'insertion de points de test et de points d'initialisation pour les circuits séquentiels dans un contexte de test pseudo-aléatoire.

Dans le Chapitre 2, nous étudierons le processus d'initialisation dans les circuits séquentiels. Nous proposerons un modèle basé sur des chaînes de Markov modifiées pour les modéliser. Ceci nous permettra de montrer que l'initialisation avec des vecteurs pseudo-aléatoires est faisable dans plusieurs cas. Pour les circuits résistants à l'initialisation par des vecteurs pseudo-aléatoires, nous développerons des heuristiques de remise à zéro partielle afin de les transformer en circuits faciles à initialiser avec des vecteurs pseudo-aléatoires.

Le chapitre 3 décrit une nouvelle mesure de testabilité dite mobilité. Nous montrerons que la mobilité est en mesure de couvrir des problèmes de testabilité séquentiels qui ne sont pas couverts par les mesures de testabilité classiques. Nous insistons ici qu'il ne s'agit pas de limitations reliées aux hypothèses d'indépendance souvent utilisées dans les mesures de testabilité classiques, mais plutôt de limitations reliées à la nature séquentielle des circuits analysés. La mobilité est basée sur le concept de probabilité de transition.

Dans un contexte de test pseudo-aléatoire où l'activité des noeuds internes est souvent anormalement élevée, l'estimation de la puissance dissipée devient importante. En effet, avec un échauffement excessif, nous risquons de brûler un composant durant le mode test. Au chapitre 3, nous développerons également une méthode basée sur la mobilité pour estimer la dissipation de puissance dans les circuits CMOS séquentiels.

Au chapitre 4, nous aborderons le problème de l'insertion des points de test dans les des circuits séquentiels. En utilisant la mobilité et une analyse probabiliste des problèmes de detection, nous développerons un ensemble d'heuristiques d'insertion, capables de transformer les circuits séquentiels résistants aux vecteurs pseudo-aleatoires en circuits faciles à tester par des vecteurs pseudo-aléatoires.

ABSTRACT

In this thesis, we will present, study and characterize new methods for test points insertion and initialization points insertion for non-full scan sequential circuits in a pseudo-random testing context.

In Chapter 2, we will study the initialization process of sequential circuits. We will propose a new model based on modified Markov chains processes to model sequential circuits behavior. Using this model, we will show that initialization with pseudo-random vectors is feasible. For those circuits which resist to the initialization with pseudo-random vectors, we will develop new partial reset methods to transform them into easy to initialize ones.

Chapter 3 describes a new testability measure called mobility. We will show that mobility can cover several test problems of sequential circuits which are not covered by classical testability measures. Note here that the test problems considered in this thesis are different from those related to the independence assumptions often encountered with approximate testability measures. They are rather related to the sequential nature of sequential circuits. The concept of mobility is based on the transition probabilities of the nodes.

During a pseudo-random test where the activity of the nodes is abnormally high, power dissipation estimation is important. In fact, with an excessive increase in power dissipation, we may burn a component during testing. In chapter 3, we will also show how mobility measures can be used for estimating power dissipation in CMOS sequential circuits.

In Chapter 4, we will study the problem of test points insertion in sequential circuits. Using the mobility and a probabilistic analysis of detection problems, we develop a set of heuristics to transform hard to test circuits into easy to test ones with pseudo-random vectors.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

Le besoin de tester les systèmes électroniques est une nécessité dictée par un marché compétitif, où la qualité est un facteur essentiel de la réussite. De nos jours, il est impératif d'assurer à un système électronique un niveau de qualité adéquat avant de le mettre sur le marché. De nombreux chercheurs et ingénieurs oeuvrant dans la conception des systèmes électroniques affirment qu'une détection tardive des défectuosités est de plus en plus coûteuse. La détection d'une défectuosité au niveau carte est 10 fois plus coûteuse que la détection de la même défectuosité au niveau composant; elle est encore 10 fois plus coûteuse si elle est détectée lors de l'assemblage du système; et si elle est détectée chez le client, le coût est encore plus important [115]. Par conséquent, il est avantageux de détecter une défectuosité le plus tôt possible.

Le test de production d'un circuit est une sorte d'expérience dans laquelle un circuit est stimulé et les résultats qu'il produit sont analysés pour vérifier s'il se comporte correctement. Plusieurs types de tests peuvent être appliqués à un circuit intégré. Les tests paramétriques sont souvent appliqués pour vérifier les caractéristiques électriques des circuits telles que les tensions de seuils, les courants de fuites, etc... Les tests fonctionnels sont généralement appliqués pour vérifier si un circuit fonctionne correctement (selon les spécifications fonctionnelles). Quant aux tests structurels (l'objet de la présente thèse[1]), ils sont appliqués pour tester la structure interne d'un circuit intégré.

---

1. Dans le reste de la présente thèse, le terme "test" désignera le test structurel.

La difficulté d'assurer un test adéquat pour un circuit intégré est une conséquence directe de l'évolution de la technologie des semi-conducteurs. En effet, cette dernière a rendu possible l'intégration de plusieurs centaines de milliers de portes dans la même puce [23]. Entre temps, le nombre de broches n'a augmenté que comme une fonction de la racine carrée du nombre de portes (règle de Rent) [93]. Cette situation rend de plus en plus complexe le problème du test des circuits intégrés modernes.

Le développement d'un test pour un circuit intégré consiste principalement à accomplir les trois tâches suivantes:

1. la génération des ensembles de vecteurs de test;
2. l'application de ces ensembles et l'observation de la réponse du Circuit Sous Test (CST);
3. la comparaison de la réponse reçue avec la réponse attendue (réponse du bon circuit).

Les stratégies de test présentement disponibles sont essentiellement destinées à réduire la complexité des trois tâches mentionnées ci-haut. Par exemple, les techniques de conception pour le test (CPT), tel que le balayage complet (défini plus loin), agissent sur la structure des circuits séquentiels, dans le but de réduire la complexité de la génération des ensembles de vecteurs de test et de faciliter l'observation de leurs réponses (réponses des circuits) lors de l'application de ces vecteurs. Un autre exemple intéressant sur lequel nous reviendrons un peu plus loin est l'autotest intégré "Built-In Self-Test". Avec ces techniques, les trois tâches mentionnées précédemment sont intégrées à l'intérieur du CST. Ceci, comme nous le verrons, peut avoir plusieurs caractéristiques désirées.

Dans le reste de ce chapitre, nous aborderons brièvement les notions de base de la modélisation des défectuosités dans les circuits intégrés. Par la suite, nous parlerons des différentes méthodes de test proposées dans la littérature (section 1.2). Ainsi, nous décrirons sommairement les techniques de conception pour le test (section 1.2.2) et les techni-

ques du test pseudo-aléatoire (section 1.2.3). Finalement, nous discuterons des motivations et de l'originalité des travaux présentés dans cette thèse (section 1.3).

1.1 Modélisation de pannes

Les dispositifs semi-conducteurs peuvent être sujets à différentes défectuosités. Ces défectuosités peuvent être dues à des erreurs de conception (i.e. spécifications incomplètes, inconsistantes ou violations des règles de conception) comme elles peuvent être dues à des défectuosités de fabrication ou à des défaillances physiques. Les défectuosités de fabrication ne sont pas directement attribuables à une erreur humaine, elles résultent plutôt d'une imperfection dans le processus de fabrication. On peut citer l'exemple des courts circuits et des circuits ouverts qui sont des défectuosités de fabrication très fréquentes. D'autres défectuosités sont attribuables à des erreurs d'alignement des masques, ou bien à une mauvaise encapsulation. Par contre, les défaillances physiques se produisent durant la durée de vie d'un système. Elles sont souvent dues à l'usure des composants ou à un stress introduit par l'environnement. Il arrive aussi fréquemment que la défaillance en opération normale soit le reflet d'une défectuosité qui a affaibli un composant d'une manière non-détectable, lors du test à la sortie de l'usine de fabrication.

En général, les défectuosités ne permettent pas une modélisation facile sur laquelle on pourrait baser le développement des tests. La modélisation par des modèles de pannes logiques s'est donc imposée comme un moyen convenable pour faciliter le développement des tests. En effet, avec un modèle de panne logique, le problème d'analyse des défectuosités et des défaillances devient un problème logique plutôt que physique. La complexité de la génération des vecteurs de test est alors réduite, puisque plusieurs défectuosités différentes peuvent être modélisées par une même panne. De plus, dans certains cas, le même

modèle peut être applicable à plusieurs technologies différentes, rendant ainsi le problème du test des circuits intégrés indépendant de la technologie.

En général, les modèles de pannes logiques supposent que tous les composants sont normaux, autrement dit, ils ne contiennent aucune défectuosité, et que seules leurs interconnexions peuvent être affectées. Les pannes typiques affectant les interconnexions sont les courts circuits et les circuits ouverts. Un court circuit est formé lorsque se connectent des lignes qui ne sont pas supposées l'être, tandis qu'un circuit ouvert résulte d'une connexion brisée.

Par exemple, dans plusieurs technologies, un court-circuit entre la masse ou l'alimentation et une ligne quelconque du circuit peut amener la ligne à prendre une valeur logique fixe. Autrement dit, la ligne est collée à une valeur logique 0 ou 1. Un court-circuit entre deux lignes quelconques crée presque toujours une nouvelle fonction logique. Dans certains cas, la fonction présentée par le court-circuit peut être un ET câblé (AND bridging fault) ou bien un OU câblé (OR bridging fault).

Depuis quelques décennies, plusieurs modèles de pannes ont vu le jour. Le modèle "collé-à" (stuck-at) est de loin le plus utilisé malgré ses limites. Pour ce modèle, chaque défectuosité se manifeste au niveau de la structure, par une ligne "collé-à 1 ou à 0". De plus, dans la plupart des approches au test, une seule panne à la fois est supposée être présente dans le circuit, lors du développement du test. Ceci est généralement nécessaire pour éviter une explosion de la complexité de la génération des vecteurs de test. Un test pour un circuit donné est considéré comme étant bon s'il couvre la majeure partie des pannes "collé-à". Dans le présent document, nous ne considérons que le modèle "collé-à" pour des pannes uniques.

1.2 Revue de littérature

La littérature relative au test est abondante. Les premiers travaux dans le domaine remontent aux années soixante [107]. En général, les solutions et les techniques de test développées jusqu'à présent émergent principalement de deux visions complémentaires:

1. développer des techniques de test de plus en plus sophistiquées, sans avoir à changer la structure du circuit à tester;
2. agir sur la structure du circuit pour le rendre plus facile à tester avec les méthodes de test dont nous disposons actuellement.

En réalité, ces deux visions ne sont pas contradictoires. Au contraire, tout développement dans l'une, peut affecter l'autre. Les techniques de génération de vecteurs de test, par exemple, s'inscrivent dans le cadre de la première vision. Durant plusieurs années, les chercheurs ont mis sur pied un ensemble de techniques de génération de vecteurs de test qui deviennent actuellement des outils importants dans les méthodologies de conception des systèmes électroniques modernes. Par ailleurs, avec des niveaux d'intégration de plus en plus élevés, il est souvent très difficile de résoudre les problèmes de test exclusivement avec des techniques de génération de vecteurs de test. De nombreux chercheurs ont ainsi proposé d'agir sur la structure du circuit lui-même pour en faciliter le test. C'est l'idée principale des techniques dites de conception pour le test (CPT) que nous aborderons un peu plus bas. Avant d'aller plus loin, nous introduirons dans la prochaine section les mesures de testabilité approximatives. Ces mesures ont une grande importance dans les techniques de génération, ainsi que dans les techniques CPT.

1.2.1 Les mesures de testabilité approximatives

Un des facteurs déterminants de plusieurs techniques de test est l'analyse de la testabilité. Cette dernière est un processus qui quantifie les problèmes de test dans un circuit donné. L'idée principale de l'analyse de testabilité est l'utilisation d'un ensemble de

mesures faciles à calculer. Ceci nous permet, tout en évitant la complexité des méthodes exactes, d'estimer la difficulté de tester un circuit donné et de localiser les sites et la nature des problèmes de test. L'analyse de la testabilité est essentiellement basée sur deux notions principales qui reflètent la difficulté de tester une panne [26][53]:

1. la contrôlabilité,
2. l'observabilité.

La contrôlabilité est une mesure de la difficulté à contrôler un noeud pour faire apparaître (sur ce même noeud) l'effet de la panne. L'observabilité est une mesure de la difficulté à observer cet effet sur les sorties primaires. De ce point de vue, la conception pour le test peut être considérée comme une discipline destinée à améliorer les deux facteurs précédents, dans le but de faciliter le développement d'un test.

Il existe dans la littérature deux grandes catégories de mesures de testabilité:

1. les mesures de testabilité déterministes;
2. les mesures de testabilité probabilistes.

Dans la première catégorie, nous retrouvons les travaux pionniers de Ruthman [109], Stephenson et Grason [131], et Breuer [24]. Ces travaux ont culminé, peu de temps après, par un des outils de calcul des mesures de testabilité les plus populaires: SCOAP [53][54]. Plus récemment, d'autres méthodes, basées sur SCOAP, ont vu le jour [20][104].

La procédure de calcul des nombres SCOAP est basée sur le rang logique des noeuds. En partant des entrées primaires se dirigeant vers les sorties primaires, les contrôlabilités sont calculées selon le rang de chaque noeud. Une fois que toutes les contrôlabilités sont calculées, la procédure de calcul des nombres SCOAP commence le calcul des observabilités, en partant des sorties primaires pour aller vers les entrées primaires. Pour de plus

amples détails sur le calcul des nombres SCOAP, veuillez consulter les références spécialisées [53][54].

Avec les mesures de testabilité probabilistes, on s'intéresse plutôt à la proportion des vecteurs d'entrées qui peuvent détecter une panne. Autrement dit, les mesures de testabilité probabilistes, comme leur nom l'indique, donnent une estimation de la probabilité de détection d'une panne. Les techniques de calcul des mesures de testabilité probabilistes que l'on retrouve dans la littérature peuvent être réparties en deux groupes.

1. les techniques probabilistes: basées sur l'étude probabiliste du circuit sans considération des vecteurs d'entrée;
2. les techniques statistiques: basées sur des simulations logiques d'un ensemble fini de vecteurs de test.

Une étude comparative des différentes techniques de calcul des mesures de testabilité fut publiée dans [61]. Les techniques statistiques semblent être efficaces si nous assurons un nombre de vecteurs de test adéquat. Malheureusement, les simulations logiques, qu'utilisent les techniques statistiques, sont relativement coûteuses et chaque vecteur de test additionnel contribuera à augmenter ces coûts. De plus, la précision de calcul dépend fortement du nombre d'échantillons (vecteurs) utilisés. Les pannes difficiles à tester, qui généralement exigent un grand nombre de vecteurs de test, peuvent complètement échapper à la détection par les techniques statistiques.

Les techniques probabilistes sont essentiellement basées sur des considérations probabilistes [44][64][116][126]. Historiquement, ces mesures furent développées pour les circuits combinatoires. Par exemple, les nombres COP [27][28], une des premières méthodes proposées pour les mesures de testabilité probabilistes ne supportent pas les circuits séquentiels. Compte tenu de ce fait, Jain et Agrawal [64] ont proposé une méthode de calcul des mesures de testabilité probabilistes qui couvre entre autre les circuits séquentiels.

Cette méthode est basée sur des statistiques collectées à partir d'une simulation logique et sur des concepts de propagation des observabilités, semblables à ceux qui sous-tendent COP. Récemment, d'autres mesures purement probabilistes pour les circuits séquentiels ont vu le jour [17][75][129]. Le chapitre 2 constitue, entre autre, une contribution aux mesures de testabilité pour les circuits séquentiels. Nous reviendrons donc sur ce sujet avec plus de détails au chapitre 2.

Les nombres COP sont principalement 3 nombres qui peuvent être calculés pour chaque noeud "N". De la même manière qu'avec la procédure de calcul des nombres SCOAP, la procédure de calcul des nombres COP traverse le circuit en partant des entrées primaires vers les sorties primaires pour calculer d'abord les contrôlabilités de tous les noeuds. Pour chaque type de porte, nous possédons une formule qui détermine les contrôlabilités de sa sortie en fonction de celles de ses entrées. Les observabilités sont ensuite calculées en allant vers les entrées primaires à partir des sorties primaires. De même, pour chaque type de porte, il existe une formule, qui pour chacune des entrées, donne son observabilité en fonction de l'observabilité de sa sortie et des contrôlabilités des autres entrées (de la même porte).

La précision des mesures de testabilité a fait l'objet de plusieurs publications. Ainsi, dans [10][88], les auteurs ont montré le caractère approximatif des nombres SCOAP. De même pour les mesures probabilistes, des chercheurs [48][61][117] ont montré que les hypothèses d'indépendance entre les entrées d'une même porte et entre l'observabilité et les contrôlabilités sont à la base d'une perte de précision des mesures COP. Ainsi, en présence de sortance reconvergeantes, ces mesures peuvent complètement échapper certains sites de problèmes de test. Par contre, ces hypothèses s'avèrent très utiles pour garder la

complexité du calcul des nombres COP linéaire en fonction du nombre de portes d'un circuit.

1.2.2 Conception pour le test

Les techniques de conception pour le test sont des techniques qui visent à tenir compte très tôt des problèmes de test durant le cycle de design. Avec l'avènement des outils de synthèse, plusieurs techniques pour faciliter le test du produit synthétisé ont vu le jour. Bien que ces techniques peuvent être considérées comme des techniques de conception pour le test, le terme "synthèse pour le test" est souvent utilisé. Dans ce qui suit, nous ne nous attarderons qu'aux techniques dites de conception pour le test, excluant celles dites techniques de synthèses pour le test.

Les techniques de CPT qui existent dans la littérature peuvent être divisées en deux groupes:

1. les techniques structurées;
2. les techniques ad hoc.

1.2.2.1 Les techniques de test structuré

Si nous faisons abstraction des techniques qui visent à pallier aux limites de précision des modèles de pannes, les techniques de test structurées développées récemment sont essentiellement destinées aux circuits séquentiels. En effet, l'état de l'art des techniques de test des circuits combinatoires est relativement mature [35]. Cependant, ceci n'est pas le cas avec les circuits séquentiels où le test introduit des problèmes reliés à l'existence des éléments de mémoire et des lignes de rétroaction [4].

Pour tester un circuit séquentiel, nous devons:

1. initialiser le circuit en contrôlant les éléments de mémoire; sans l'initialisation, la réponse du circuit ne peut être prédite avec certitude;
2. appliquer une séquence de plusieurs vecteurs de test.

Pour détecter une panne dans un circuit séquentiel, nous devons d'abord amener le circuit à travers son diagramme de transition d'états, jusqu'à un état où la panne en question est excitée, ensuite nous devons assurer la propagation de son effet (de la panne), toujours à travers le diagramme de transition, jusqu'à une sortie primaire. Donc, en plus des problèmes de dépendance structurelle entre les noeuds, nous devons tenir compte des dépendances temporelles dans le cas des circuits séquentiels. Dans ce qui suit, nous ne mentionnerons que deux approches représentatives des techniques CPT.

Balayage complet. Il y a plus de deux décennies, plusieurs chercheurs [13][43][47] ont proposé de convertir le problème de test des circuits séquentiels en un problème de test des circuits combinatoires. L'idée principale sous-jacente à ces techniques est de transformer les éléments de mémoires, dans le but de les relier entre eux dans une (ou plusieurs) longue(s) chaîne(s) de balayage durant le mode test. De cette manière, tous les éléments de mémoire deviennent directement contrôlables et observables, via la chaîne de balayage.

Ces techniques, connues sous le nom de balayage complet, sont très répandues. Actuellement, elles sont devenues la norme dans l'industrie des dispositifs à semi-conducteurs; et ceci malgré la résistance de certains concepteurs [19].

Cette résistance émane en réalité de certains inconvénients que peut poser l'adoption de telles techniques. Parmi ces inconvénients, nous citons sans être exhaustif:

1. la circuiterie ajoutée, pour pouvoir transformer les éléments de mémoire en chaîne de balayage complète durant le "mode test";
2. la dégradation de la performance encourue par l'ajout de circuiterie. Ceci est particulièrement problématique dans le cas des applications à haute performance où les chemins critiques sont nombreux;

3. l'augmentation du temps d'application du test, car nous devons charger la chaîne de balayage à chaque fois qu'un vecteur de test doit être appliqué sur la partie combinatoire du circuit, capturer la réponse du circuit et faire décaler l'information retenue dans la chaîne pour en lire son contenu;
4. la qualité du test. En effet, avec les techniques de balayage, le test est appliqué sous des conditions temporelles moins sévères que les conditions de fonctionnement normal ("*at-speed*"). Par conséquent, certains problèmes de test (qui sont peu ou pas modélisés par le modèle de pannes colle-à) peuvent échapper à la détection, affectant ainsi la qualité du test.

Balayage partiel. Pour remédier partiellement aux problèmes encourus par le balayage complet, plusieurs auteurs [133][134] ont proposé les techniques de chaîne de balayage partiel (dites balayage partiel). Avec ces techniques, seule une partie des éléments de mémoire sera sélectionnée pour former une chaîne de balayage partiel. Ainsi, la circuiterie ajoutée est réduite, la dégradation des performances peut être améliorée en évitant les chemins critiques, et finalement le temps d'application de chaque vecteur de test sera réduit.

En réalité, une comparaison des techniques de balayage partiel avec celles du balayage complet n'est pas aussi simple, tel que nous l'avons présenté ci-haut. Plusieurs auteurs remettent en cause les bienfaits du balayage partiel. En effet, le problème de la génération algorithmique demeure dans ce cas, essentiellement séquentiel. Par conséquent, le gain au niveau de la circuiterie, ou au niveau du temps d'application, peut facilement être annulé ou même devenir une perte nette à cause de la complexité de la génération algorithmique et de la perte en couverture de pannes.

De toute façon, le balayage partiel a fait l'objet d'un très grand nombre de publications comme en témoigne la liste bibliographique ci-jointe [8][9][34][35][36][37][38][39][57][69]. Un problème relié au balayage partiel consiste à choisir parmi un grand nombre d'éléments de mémoire d'un circuit, un sous-ensemble qui sera inclus dans une

chaîne de balayage. Ceci a pour but de garantir un certain niveau de test (exprimé souvent en pourcentage de pannes détectées) tout en minimisant les inconvénients qui en résultent.

Les techniques de choix des éléments de mémoire qui existent dans la littérature, peuvent être divisées en trois catégories:

1. les techniques basées sur les mesures de testabilité [133];
2. les techniques basées sur la génération algorithmique [36][38];
3. les techniques structurelles [35][57];

L'idée principale de la 1ère catégorie est d'extraire un ensemble de paramètres à partir d'un ensemble de mesures de testabilité. Ces paramètres serviront à déterminer un petit ensemble de bascules pour former une chaîne de balayage partiel. Avec les techniques de la 2ième catégorie, un générateur de vecteurs de test algorithmique est utilisé. À partir de l'expérience acquise durant la génération algorithmique, un petit ensemble de bascules est choisi pour couvrir les pannes restantes. Finalement, avec les techniques structurelles, les bascules sont choisies pour former la chaîne de balayage partiel, suivant leurs contributions pour réduire l'impact des rétroactions et la profondeur séquentielle des circuits.

1.2.2.2 Insertion de points de test

Une approche Ad Hoc directe pour améliorer la testabilité est l'insertion de points de test. Historiquement, cette approche fut principalement développée pour des circuits combinatoires [73][112][119][132][145]. Ceci est probablement dû au fait que les techniques structurées ne peuvent pas résoudre tous les problèmes de tests. Des dépendances logiques dans la partie combinatoire d'un circuit sont parfois insensibles ou peu sensibles à la chaîne de balayage. Récemment, d'autres travaux d'insertion de points de test destinés aux circuits séquentiels ont vu le jour [75][90][91]. Dans ces travaux, on reconnaît le fait qu'une bascule peut ne pas être le meilleur endroit pour l'insertion. Le chapitre 4 consti-

tue, entre autre, une contribution au domaine de l'insertion des points de test dans les circuits séquentiels.

Une formulation du problème d'insertion des points de test consiste à choisir un petit ensemble de points pour insérer des structures logiques additionnelles (voir Figure 1.1), tout en:

1. minimisant le nombre de ports d'entrée/sortie additionnels;
2. minimisant l'augmentation de la surface globale occupée par la circuiterie ajoutée;
3. limitant les dégradations de la performance;
4. facilitant le développement du test.

Notons que les quatre points mentionnés ci-haut ne sont pas nécessairement de même priorité. En effet, l'objectif principal de toutes les techniques d'insertion de points de test est de garantir avant tout un bon niveau de testabilité (exprimé généralement en couverture de pannes). Evidemment, ceci doit s'effectuer tout en respectant le budget alloué en termes de surface additionnelle permise, de nombre de ports supplémentaires possibles et de dégradation de performance tolérée.

Ici, nous pouvons déjà distinguer deux catégories de points de test, et par conséquent deux sortes de logiques additionnelles pour résoudre deux sortes de problèmes de test différents. Les deux catégories sont:

1. les points de contrôle qui sont destinés à améliorer la contrôlabilité des noeuds internes et ainsi résoudre ce qu'on appelle les problèmes de contrôle;
2. les points d'observation dont le rôle est d'améliorer l'observabilité des noeuds internes et ainsi régler les problèmes connus sous le nom de problèmes d'observabilité.

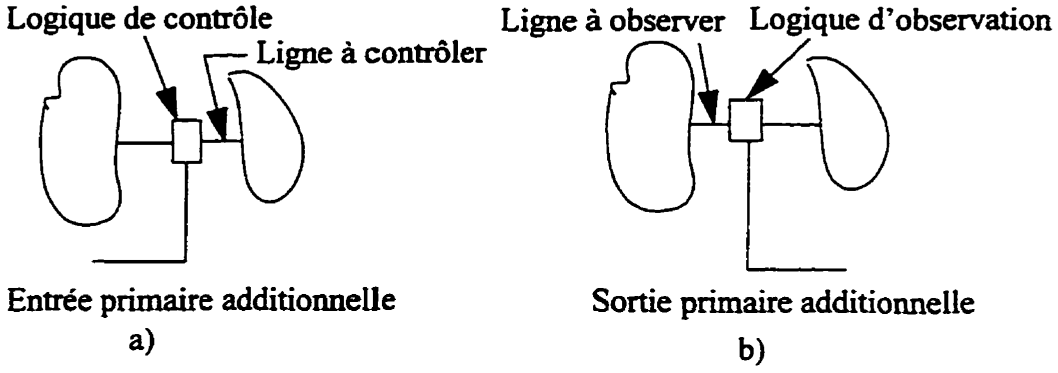


Figure 1.1 Les deux catégories de points de test a) point de contrôle b) point d'observation

Dans les travaux reliés à l'insertion des points de test, nous pouvons distinguer quatre notions fondamentales:

1. la structure des points de test, autrement dit, la forme de la logique utilisée pour le point de test;
2. l'emplacement de ces points dans le réseau des portes d'un circuit donné;
3. la condensation de ces points;
4. les sources pseudo-aléatoires utilisées.

Structure des points de test. L'importance des structures de points de test découle évidemment du fait de vouloir réduire les pénalités encourues par l'insertion, tout en garantissant un bon niveau de testabilité. En effet, à chaque insertion de point de test peut correspondre une "circuiterie ajoutée" et une dégradation des performances. Parmi les structures proposées dans la littérature, nous citons sans être exhaustif:

1. les points de test classiques: Une porte AND est utilisée pour contrôler un noeud à 0, une porte OR pour le contrôler à 1 et une ligne vers l'extérieur pour l'observer (voir Figure 1.2);
2. les points FO pour la technologie CMOS (voir Figure 1.3);
3. les portes XOR;

4. cellule de test (Test-cell) (voir Figure 1.4).

Les points de test classiques furent largement utilisés dans le domaine de l'insertion des points de test. Cependant, ce genre de points est généralement associé à une augmentation de la circuiterie ajoutée et à une dégradation de la vitesse du circuit non négligeables. De plus, l'insertion d'un point classique coupe une ligne en deux parties, la contrôlabilité de la partie reliée à la sortie de la porte insérée est améliorée, par contre, celle de la partie reliée à l'entrée de la porte demeure aussi mauvaise qu'avant l'insertion. Ce problème est connu sous le nom de "dédoublement de la personnalité" [15].

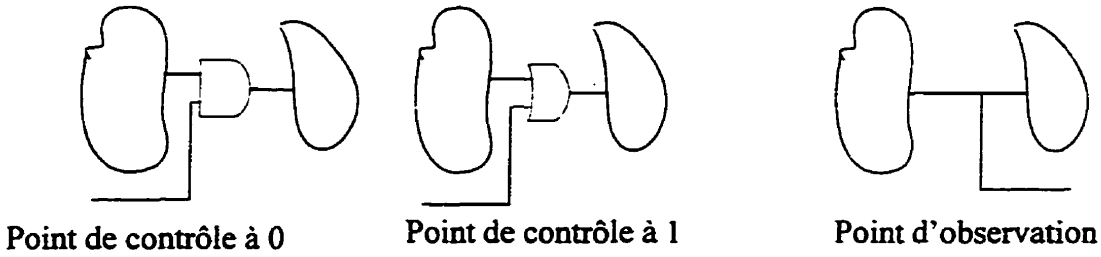


Figure 1.2 Les points de test classiques

L'approche FO (Force-Observe) [113][114] fut développée spécifiquement pour la technologie CMOS. Avec cette approche, les portes correspondantes aux lignes à contrôlabilité faible sont remplacées par des portes complexes tel que montré sur la Figure 1.3. Ceci assure un effet similaire à celui d'une porte AND/OR sur la testabilité, tout en réduisant la circuiterie ajoutée et la dégradation des performances. De plus, le problème de "dédoublement de la personnalité" est éliminé avec l'approche FO. Le deuxième élément de cette approche est la capacité d'observer n'importe quelle ligne par l'insertion d'un point d'observation de faible coût (voir Figure 1.3). Pour plus de détails sur l'approche FO, référez-vous aux publications [113][114].

Les portes XOR présentent des caractéristiques très appréciées comme points de contrôle. En effet, les portes XOR sont de bons régulateurs de contrôlabilité: quelque soit la contrôlabilité d'une ligne, l'insertion d'une porte XOR améliore de façon équilibrée ses contrôlabilités à 0 et 1. Plus encore, il n'y a généralement pas de dégradation de l'observabilité du cône d'entrée de cette ligne. Par exemple, selon COP, l'observabilité d'une entrée d'une porte XOR est égale à l'observabilité de sa sortie. Nous analyserons avec plus de détails l'effet de la porte XOR sur la testabilité d'un circuit au chapitre 4.

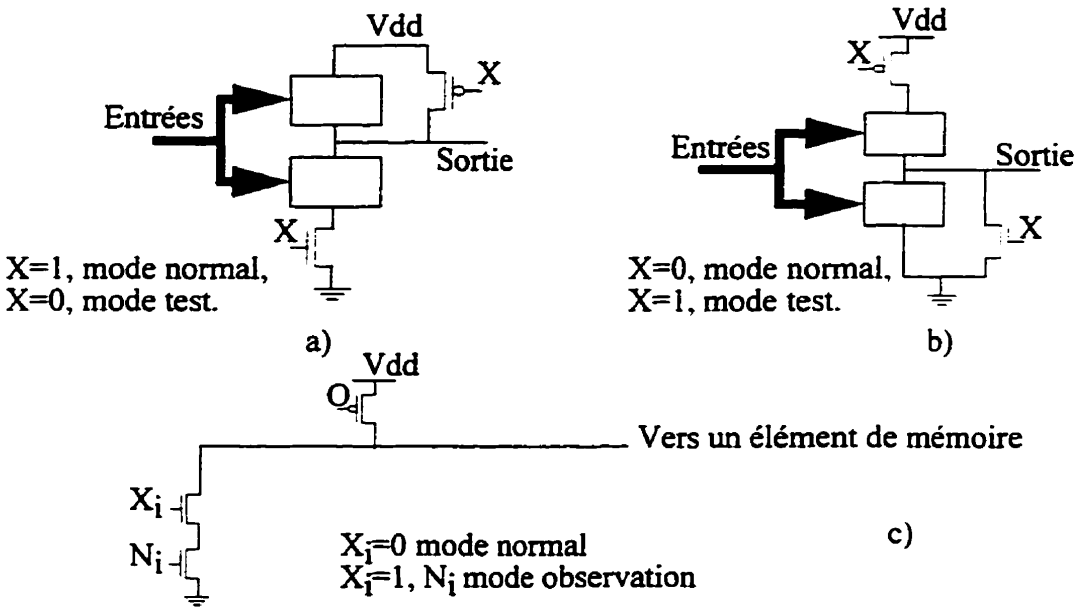


Figure 1.3 Points de test FO. a) contrôle à 1, b) contrôle à 0, c) observation

L'approche "cellules de test" (Test-cell) s'inspire des techniques de balayage. Nous pouvons même considérer les techniques de balayage, dans une certaine mesure, comme un cas particulier de cette approche. Une cellule de test est une bascule avec des mécanismes de balayage. L'insertion peut se faire sur n'importe quelle ligne, elle n'est pas limitée

aux éléments de mémoire comme dans le cas des techniques de balayage. En mode test, ces bascules sont reliées entre elles pour former une (ou plusieurs) chaîne(s) de balayage (voir Figure 1.4).

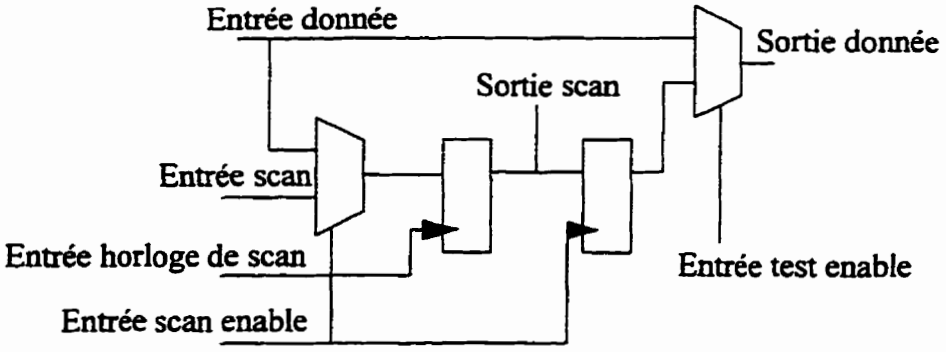


Figure 1.4 Une cellule de test (T-cell)

Sélection des points de test. Le problème de la sélection des points de test est reconnu comme étant un problème NP-complet (pour les circuits avec reconvergences) [63]. Dans la littérature, nous pouvons énumérer trois catégories de techniques de sélection:

1. les techniques de sélection pour simplifier la génération algorithmique [56][60][62] [70][71];
2. l'insertion de points de test pour "segmenter" ou "partitionner" un circuit, dans le but de faciliter son test pseudo-exhaustif;
3. l'insertion de points de test pour améliorer le test pseudo-aléatoire [25][66][84][134].

Dans la première catégorie, le but de l'insertion est de faciliter la génération algorithmique. Durant la dernière décennie, plusieurs travaux de cette catégorie furent proposés. Nous citons ici le travail de Gundlach [56], où l'insertion consiste à trouver un petit ensemble de points pour couper les rétroactions et réduire les reconvergences. Ainsi, la génération algorithmique est améliorée du moment où les deux facteurs qui influencent sa

complexité (rétroactions et reconvergences) sont éliminés. Kim et al [70] présentent une autre méthode d'insertion, basée sur des informations extraites des dessins de masques et de la génération algorithmique, pour choisir un ensemble de points d'observation seulement. Avec cette technique, l'insertion s'effectue au niveau du dessin des masques afin d'en minimiser le coût associé. Une technique similaire, basée sur l'expérience acquise durant la (ou les) génération(s) algorithmique(s) précédente(s), est disponible avec l'outil de génération algorithmique commercial Intelligen [62].

La deuxième catégorie englobe toutes les techniques destinées à rendre pratique le test exhaustif. Nous y reviendrons dans la prochaine section, lorsque nous parlerons des techniques de test aléatoire.

La dernière catégorie est destinée à améliorer le test pseudo-aléatoire qui trouve évidemment une application directe dans les techniques d'autotest intégrées (BIST). Dans cette catégorie, nous pouvons distinguer deux grandes sous-catégories:

1. les techniques basées sur la simulation de pannes [25][63];
2. les techniques basées sur les mesures de testabilité [119][132][145];

Dans la première sous-catégorie, nous pouvons citer le travail de Briers et al. [25]. Dans leur article, ils ont proposé une heuristique de placement pour éliminer les pannes résistantes au test pseudo-aléatoire. Cette heuristique est basée sur deux simulations de pannes, dont une première pour déterminer les noeuds ayant une contrôlabilité faible. Les points insérés durant cette phase sont destinés à éliminer la dépendance entre les noeuds d'une même porte. La deuxième simulation sert à améliorer les observabilités. Un second travail dans cette catégorie fut proposé par Iyengar et Brand [63]. Avec leur heuristique, une simulation de panne est invoquée pour ramasser des informations sur les problèmes de propagation que rencontrent les pannes résistantes au test pseudo-aléatoire dans un réseau

de portes. Les points de contrôle sont choisis selon leur contribution à améliorer la propagation de l'effet de ces pannes. Une deuxième simulation est ensuite invoquée pour déterminer l'ensemble des points d'observation.

Evidemment, ces techniques sont relativement coûteuses, car elles se basent sur la simulation de pannes. De plus, la simulation de pannes est étroitement reliée à la séquence de vecteurs de test utilisée. Bien entendu, ceci limite la précision de ce genre de techniques. Notons cependant à ce niveau, qu'Iyengar et Brand [63] ont déjà mentionné ce problème. Ainsi, ils ont montré avec des résultats expérimentaux que la fluctuation de la couverture de pannes, en fonction de la séquence de test appliquée, est de l'ordre de 1% seulement.

Pour éliminer le recours à la simulation de pannes, plusieurs auteurs ont proposé des méthodes heuristiques basées sur les mesures de testabilité. Evidemment, ceci est fait au détriment de la précision. Cependant, malgré cette imprécision, ces techniques ont prouvé leur efficacité et elles sont actuellement largement utilisées, particulièrement dans le test des circuits combinatoires. Par exemple, Youssef et al. [145] ont proposé des heuristiques de placement qu'ils ont implantées dans un outil de conception pour le test pseudo-aléatoire. Leurs heuristiques sont basées sur les nombres COP, les notions de secteurs de pannes ainsi que la corrélation entre les différents points de test. Un ensemble de points résistants au test pseudo-aléatoire est déterminé en premier lieu par le biais des nombres COP. Ces points sont ensuite regroupés dans des secteurs sous forme de cônes, dits secteurs de pannes. L'insertion s'effectue au niveau des sommets de ces secteurs seulement. À l'aide de FO+, Youssef et al. ont montré qu'un petit ensemble de points de test est généralement suffisant pour éliminer toute résistance au test pseudo-aléatoire dans le cas des circuits combinatoires. Seiss et al. [119] ont proposé de leur coté une technique similaire

basée sur une fonction de coût qui reflète la testabilité pseudo-aléatoire des circuits combinatoires. Récemment, Tamarapalli et Rajski [132] ont publié un article sur une technique d'insertion de points de test dite multiphase. Dans chaque phase, un sous-ensemble de pannes est considéré. Les points de contrôle de chaque phase sont activés par des valeurs fixes.

Les techniques de condensation des points de test. Il est connu dans le domaine du VLSI que le nombre de ports d'entrée/sortie est une ressource limitée. Par conséquent, une bonne méthode de test doit composer avec cette réalité. Par exemple, avec les techniques de balayage, telle que l'approche LSSD, seulement 4 ports d'entrée/sortie sont utilisés. Cette limitation relative au nombre de ports a motivé le développement des techniques de condensation dans le domaine de l'insertion des points de test. La condensation des points de test consiste à diviser l'ensemble de ces points en plusieurs groupes, de manière à ce que le nombre de ports d'entrée/sortie additionnel dont nous avons besoin pour les contrôler soit minimal. Dans la littérature, nous pouvons distinguer quatre classes de techniques de condensation:

1. la chaîne de balayage;
2. l'arbre de portes XOR;
3. le multiplexage;
4. les sources d'excitations pseudo-aléatoires internes.

Dans le cas où les points de test sont des bascules (T-cell par exemple), nous pouvons les relier entre elles dans une chaîne de balayage avec un petit nombre de ports d'entrée/sortie. Ainsi, nous pouvons injecter les données dans la chaîne de balayage de façon sérielle et lire leurs contenus sur une sortie primaire.

D'autres auteurs ont proposé de condenser les points d'observation avec des arbres de portes XOR. La racine de chaque arbre est connectée avec une broche de sortie. Cette

technique n'est pas à l'abri de certains problèmes reliés aux pertes d'informations. Notamment, si les effets d'une ou de plusieurs pannes se propagent aux entrées d'une même porte XOR, leurs effets s'éliminent mutuellement. Ce phénomène est connu sous le nom de "aliasing".

Le multiplexage est également utilisé pour réduire le nombre de broches d'entrée/sortie nécessaires. Avec cette méthode, les broches fonctionnelles (utilisées pour accomplir la mission du circuit ou les bascules reliées dans une chaîne de balayage) sont multiplexées avec les points de test durant le mode test. Dans le cas où nous ne pouvons pas utiliser tous les points de test en une seule fois par manque de broches d'entrée/sortie, il est toujours possible de définir plusieurs modes de test. Dans chaque mode, un sous-ensemble des points de test est exploité. Par exemple, dans l'approche FO, Youssef et al. [145] ont proposé un mécanisme de condensation des points d'observation en plusieurs modes. Avec ce mécanisme, plusieurs points d'observation sont reliés à une même ligne. Chaque ligne est, à son tour, reliée dans la mesure du possible à un élément de mémoire. Ainsi, l'ensemble des points de test est divisé en plusieurs sous-ensembles et le nombre de broches additionnelles nécessaires est minimal. Durant le mode test, les mêmes auteurs proposent de se restreindre à n'observer qu'un seul point par groupe à la fois. Par conséquent, le nombre maximal de modes d'observations est déterminé par le nombre de points dans le plus grand sous-ensemble (voir Figure 1.5 pour une illustration de la condensation des points d'observation avec l'approche FO)

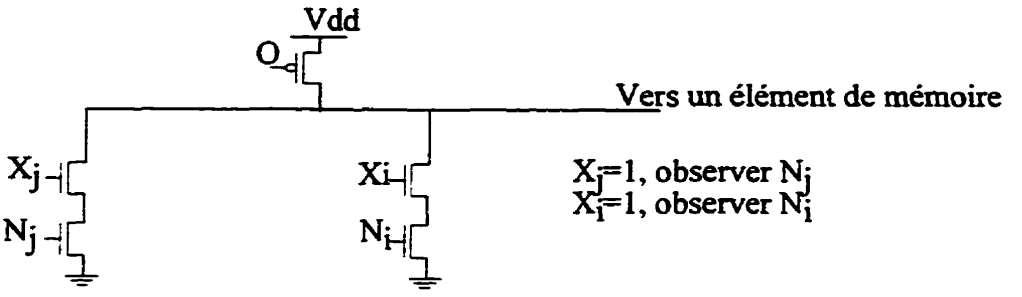


Figure 1.5 Condensation des points de test avec l'approche FO

Récemment, Soufi et al. ont avancé l'idée d'utiliser des points internes comme sources pseudo-aléatoires dans le but d'alimenter les lignes set/reset d'un reset partiel [126]. Ainsi, le recours à des sources externes est minimisé, et la taille du générateur pseudo-aléatoire est réduite. Cette idée fut également exploitée par Muradali et al. [90] pour alimenter des points de test par des sources pseudo-aléatoires internes.

1.2.3 Conception pour l'autotest intégré "Built-In Self-Test"

Avec la conception pour l'autotest intégré, un circuit, une puce, une plaque ou un système peut se tester (lui-même) sans aucun apport de l'extérieur. Ceci a pour objectif:

1. d'éliminer le besoin d'avoir des équipements de test qui sont généralement coûteux (un million de $ pour un testeur est un prix courant);
2. de réduire le volume des données (vecteurs de test) qui doit être manipulé par les équipements de test (vecteurs de test et réponses du CST);
3. de réduire la complexité de la génération et de l'application des ensembles de vecteurs de test;
4. et finalement la réutilisation d'un test (test reuse) développé à un ou des niveaux d'abstraction supérieurs.

En général, les approches BIST peuvent être caractérisées par la nature de la génération utilisée et par la forme du test appliqué. Ainsi, on parle de BIST algorithmique versus pseudo-aléatoire et de BIST exhaustif versus non exhaustif.

Le test exhaustif garantit toujours 100% de couverture des pannes "collé-à" testables. Les techniques du test exhaustif n'ont besoin ni d'un modèle de pannes, ni d'une simulation de pannes pour valider les résultats. Cependant, elles requièrent la possibilité de visiter tous les états ainsi que la possibilité de faire toutes les transitions possibles (entre les paires d'états). Ceci n'est en général pas faisable pour des circuits de complexité VLSI.

Dans le but de réduire la complexité des techniques de test exhaustif, de même que d'en préserver les avantages, plus spécifiquement le 100% (ou presque) de couverture de pannes, plusieurs auteurs ont proposé des approches hybrides dites pseudo-exhaustives. Ces techniques consistent généralement à partitionner un circuit donné sous forme de sous-blocs, de manière à réduire le nombre des entrées de cônes d'entrée des sorties primaires. La littérature présente de nombreux travaux dans cette catégorie. Pour plus d'informations veuillez vous référer aux publications spécialisées sur le sujet [12][65][85][142].

Avec le test pseudo-aléatoire, un ensemble de vecteurs aléatoirement générés est utilisé comme ensemble de vecteurs de test. À l'opposé des techniques exhaustives, le test pseudo-aléatoire a l'avantage d'être réalisable pour des circuits combinatoires ou séquentiels pratiques. Les vecteurs de test, dans ce cas, sont indépendants de la fonction du système. Souvent, des circuits très simples (LFSR[1], CA[2]) sont utilisés comme générateurs pseudo-aléatoires. Malheureusement, l'estimation de la couverture de pannes et la qualité

1. LFSR: de l'anglais Linear Feedback Shift Register
2. CA: de l'anglais Cellular Automata

des tests demeurent encore problématiques avec les techniques pseudo-aléatoires. En effet, les ensembles de vecteurs de test pseudo-aléatoires sont généralement plus longs que ceux obtenus par une génération algorithmique. Par conséquent, estimer la couverture de pannes par le biais de la simulation de pannes, comme nous le faisons habituellement, peut être coûteux. Dans le but de surmonter ce problème, des techniques analytiques furent proposées dans la littérature. Ces techniques ne sont que des approximations et aucune d'entre elles n'est en mesure de donner une valeur exacte de la couverture de pannes. Le $2^{ième}$ problème qui est probablement le plus important, est d'amener la couverture de pannes à un niveau acceptable. En effet, des études ont montré que dans plusieurs cas, le niveau de testabilité (couverture de pannes) atteint n'est pas acceptable. Les chercheurs ont observé dans plusieurs circuits des pannes difficiles à tester avec des vecteurs pseudo-aléatoires. Ce genre de pannes est connu sous le nom de pannes résistantes au test pseudo-aléatoire. Un exemple simple de pannes résistantes est la sortie collée à 0 d'une porte AND avec "n" entrées.

Plusieurs approches furent proposées dans la littérature pour faire face au problème des pannes résistantes. Ces approches peuvent être classifiées en trois groupes:

1. les techniques basées sur la génération algorithmique;
2. les techniques du test pondéré;
3. les techniques d'insertion de points de test.

Avec les techniques basées sur la génération algorithmique, un générateur algorithmique est utilisé afin de produire un ensemble de vecteurs de test pour les pannes résistantes. Cet ensemble est ensuite stocké dans une ROM [5]. Durant la phase test, les vecteurs de test de cet ensemble sont appliqués en premier lieu, suivis d'autres vecteurs générés aléatoirement. Dans d'autres techniques de la même catégorie, le générateur algorithmique est

utilisé pour générer un test complet, et par la suite, un générateur pseudo-aléatoire est utilisé pour produire un ensemble contenant le test algorithmique déjà généré [21].

Avec les techniques de la $2^{ième}$ catégorie, des vecteurs pondérés de façon non uniforme sont utilisés [77][92][141]. Malheureusement, certaines structures peuvent résister même aux tests pondérés [143]. Un exemple d'une telle situation est montré à la Figure 1.6. Un seul ensemble de poids n'est pas suffisant dans ce cas. Nous devons générer plusieurs ensembles de poids pour faire face à une telle situation. Le principe sous-jacent aux techniques utilisant des ensembles de poids multiples est principalement basé sur le partitionnement de l'ensemble des pannes d'un circuit en plusieurs sous-ensembles. Chaque sous-ensemble peut être couvert par un seul ensemble de poids. D'autres auteurs ont proposé une méthode de partitionnement indirect, où l'ensemble des vecteurs de test algorithmiquement généré est partitionné, à la place de l'ensemble des pannes.

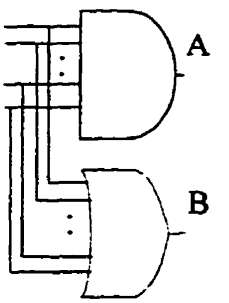


Figure 1.6 Un exemple d'un circuit résistant à un test pondéré avec un seul ensemble de poids

1.3 Objectifs de la thèse

Les objectifs de cette thèse sont d'étudier, d'analyser et de développer des outils rapides pour tester les circuits séquentiels dans un contexte de test pseudo-aléatoire. Ce travail vise à développer de nouvelles méthodes plus raffinées et plus efficaces pour l'insertion de points de test dans des circuits séquentiels sans chaîne de balayage. L'originalité de la présente thèse réside à notre avis à plusieurs niveaux. Tout d'abord, nous avons proposé

un modèle de chaîne de Markov modifié pour modéliser les circuits séquentiels. Ce modèle nous a permis de mieux comprendre les mécanismes qui conduisent à l'initialisation des circuits séquentiels dans un contexte de test pseudo-aléatoire et ainsi de démontrer la faisabilité du processus. Ceci nous a également permis de proposer des méthodes de reset partiel, pour rendre un circuit séquentiel facile à initialiser. Cette contribution fera l'objet du chapitre 2. Deuxièmement, nous avons montré quelques limites des mesures de testabilité classiques qui influencent le test des circuits séquentiels. Ainsi, nous avons proposé la mobilité comme mesure pour tenir compte de ces limites. Avec la mobilité, nous pouvons faire une estimation plus précise de la testabilité des circuits séquentiels, ainsi qu'une estimation de la dissipation de puissance, souvent nécessaire pour savoir si l'activité durant un test pseudo-aléatoire est tolérable. La mobilité, ainsi que les limites des mesures de testabilité classiques, feront l'objet du chapitre 3. Une autre contribution de cette thèse est l'utilisation du concept de la mobilité pour faire de l'insertion de points de test. Nous avons développé un ensemble d'heuristiques d'insertion de points de test, où nous tenons compte de l'effet conjoint des groupes de points de test. Cet effet peut améliorer le processus d'insertion comme il peut être néfaste dans certains cas. Des modes de test différentes, invoqués en séquence, sont proposés comme solution à ces problèmes. Les heuristiques d'insertion ainsi que l'analyse de la testabilité des circuits séquentiels feront l'objet du chapitre 4. Les conclusions qui découlent de cette thèse seront présentées au chapitre 5 qui en rappelle les points saillants et les axes de recherches futures possibles.

CHAPITRE 2

Initialisation des circuits séquentiels avec des vecteurs pseudo-aléatoires

Le problème de test des circuits séquentiels nécessite une phase d'initialisation. Si un circuit n'est pas complètement initialisé, des pannes peuvent échapper à la détection malgré qu'elles soient faciles à détecter lorsque le circuit est initialisé. Dans un contexte de test pseudo-aléatoire, l'initialisation est aussi importante. Un seul bit non initialisé est suffisant pour corrompre la bonne signature attendue avec n'importe quel analyseur de signatures, résultant ainsi dans des pertes de couverture. L'initialisation est le processus qui conduit un circuit à un état initial connu à partir d'un état complètement inconnu. Dans certains cas, elle peut être très difficile, voire même impossible à obtenir, et elle peut exiger des longueurs de séquence importantes [22].

Avec les techniques de test pseudo-aléatoire, une initialisation déterministe est souvent requise. Cette dernière est généralement obtenue avec une séquence de vecteurs déterministes. De plus, une circuiterie additionnelle dédiée à l'application de cette séquence est souvent nécessaire. Normalement, le test pseudo-aléatoire ne débute qu'après avoir appliqué la séquence d'initialisation déterministe.

Dans le présent chapitre, nous étudierons le problème d'initialisation des circuits séquentiels dans un contexte de test pseudo-aléatoire. Après une brève introduction où une revue de littérature sur le sujet est présentée, nous aborderons à la section 2.2 une étude

sur le coût de la technique d'initialisation dite de *reset* complet, qui est largement utilisée. Un résumé de nos recherches présenté dans un article sur l'initialisation pseudo-aléatoire des circuits séquentiels sera présenté à la section 2.3. L'article lui-même fera l'objet de la section 2.4. Nous présenterons dans la section 2.5 une technique d'initialisation dite de *reset* partiel (par opposition aux techniques de *reset* complet). Cette technique transforme les circuits résistants à l'initialisation par des vecteurs pseudo-aléatoires en circuits faciles à initialiser. Dans la section 2.6, nous discuterons de l'effet du *reset* partiel sur la testabilité. Nous proposerons également d'autres techniques de *reset* partiel pour améliorer l'initialisation et la testabilité.

2.1 Introduction

Dans le but de garantir l'initialisation, un concepteur peut exiger de n'utiliser que des bascules avec des mécanismes de mise-à-0/mise-à-1 (entrées *set/reset*). Cette technique où toutes les bascules sont munies de mécanismes de mise-à-0/mise-à-1 est connue sous le nom de *reset* complet[1]. Dans ce cas, un seul patron de test est suffisant pour déterminer l'état initial du circuit. Malheureusement, un *reset* complet exige une circuiterie additionnelle atteignant parfois 10% de la surface globale du circuit. Éventuellement, cette circuiterie additionnelle s'ajoute au coût de la chaîne de balayage.

Utiliser systématiquement des bascules avec des mécanismes de mise-à-0/mise-à-1 (*reset* complet) pour forcer une initialisation déterministe sur toutes les bascules n'est parfois pas nécessaire. Par exemple, les bascules utilisées comme éléments de délai ou bien pour la resynchronisation (*retiming*) ne contribueront pas nécessairement à un problème d'initialisation. Le *reset* complet est souvent facultatif, si le but est d'assurer l'initialisation pour des fins de test. Lin et al. [75] ont déjà reconnu ces inconvénients et ils ont pro-

1. *reset* complet = de l'anglais "full reset".

posé de remplacer seulement les bascules dites "*self-loop flip-flops*" par leurs équivalents avec des mécanismes de mise-à-0/mise-à-1. Une bascule est appelée "*self-loop flip-flop*" si et seulement si sa sortie est reliée à son entrée exclusivement à travers un bloc de logique combinatoire. Ici encore, il n'est pas évident que toutes les bascules de type "*self-loop flip-flops*" soient responsables des problèmes d'initialisation. La figure 2.7 de la page 39 présente un cas d'une bascule "*self-loop*" qui n'a pas de problème d'initialisation. Récemment, Cheng and Agrawal [33] ont mis de l'avant l'idée d'utiliser un *reset* partiel pour faciliter l'initialisation, dans le cas des machines à états finis avec (ou sans) une séquence de synchronisation. Cette idée fut exploitée par plusieurs auteurs comme nous le verrons un peu plus loin (voir section 2.5).

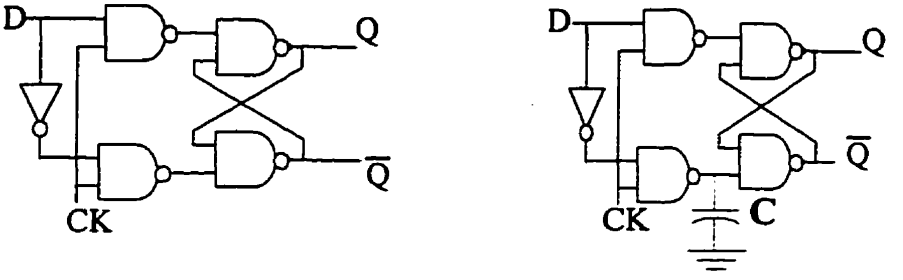


Figure 2.1 Une mise en oeuvre d'une bascule de type D en utilisant les portes NAND et son équivalent asymétrique

Une autre idée consiste à utiliser des bascules asymétriques [101] pour résoudre les problèmes d'initialisation. Aucune littérature n'existe sur ce sujet. Une bascule asymétrique est une bascule biaisée de manière à ce qu'au moment de la mise en marche "*power on*", la bascule converge naturellement vers 0 ou 1 dépendamment du biais utilisé. La figure 2.1 montre un exemple d'une bascule-D avec son équivalent asymétrique, et la figure 2.2 montre des simulations SPICE de la bascule asymétrique où l'alimentation est modélisée comme une rampe. Les résultats de ces simulations montrent que l'initialisation par des bascules asymétriques est possible (figure 2.2-a). Cependant, cette dernière (initia-

lisation) dépend fortement des charges sur Q et $\overline{Q}$. Une autre simulation SPICE (figure 2.2-b) nous montre que l'initialisation peut s'inverser si une charge suffisante est appliquée sur Q ou $\overline{Q}$. Les bascules asymétriques sont aussi sensibles au bruit lorsque des rampes d'alimentation avec de petites pentes sont utilisées. Une perturbation sur l'alimentation peut également inverser l'initialisation.

Les bascules asymétriques s'avèrent être une solution possible au problème d'initialisation. Cependant, pour les raisons techniques mentionnées plus haut et le peu d'information sur le sujet, nous ne considérons dans ce chapitre que les techniques de *reset* explicite avec des entrées de mise-à-0 / mise-à-1.

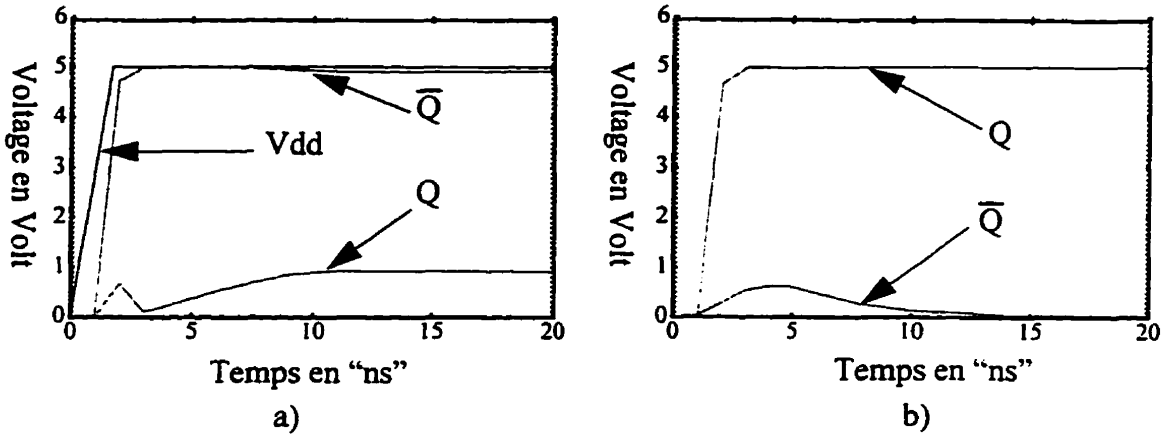


Figure 2.2 Résultats des simulations SPICE sur la bascule asymétrique montrée sur la Figure 2.1 a) sans charge sur les sorties b) avec une charge de 0.2pF sur Q.

Dans la littérature sur le problème d'initialisation, nous trouvons entre autres les travaux de Pixeley et Beihl [103], de Pixely et al. [102] et de Rho et al. [105]. Dans [103], Pixeley et Beih proposent une technique pour vérifier si une machine à états finis donnée est initialisable. Ils proposent aussi une méthode pour en extraire une séquence d'initiali-

sation dans le cas favorable. Dans [102], une autre méthode exacte pour extraire les séquences d'initialisation basée sur les BDD (Binary Decision Diagram) fut proposée. Dans [105], Rho et al. ont publié une nouvelle technique pour extraire la plus petite séquence de synchronisation d'une machine à états finis donnée. Évidemment, plusieurs séquences de synchronisation minimales peuvent exister. La technique proposée dans [105] garantit l'extraction d'au moins une séquence.

Jusqu'ici, nous n'avons mentionné que les travaux dont l'objectif est de garantir/ extraire une séquence d'initialisation pour un circuit en mode de fonctionnement normal (autrement dit sans pannes). Malheureusement, ceci ne garantit pas l'initialisation de ce même circuit en présence de certaines pannes, dites pannes qui peuvent empêcher l'initialisation. Un exemple simple de ce genre de situation apparaît lorsque le *reset* d'une boucle est collé à la valeur inverse de sa valeur active. Dans [1], Abramovici et Parikh ont proposé une méthode qui nous permet d'identifier directement ces pannes, et d'extraire si possible, une ou plusieurs séquences de vecteurs pour assurer l'initialisation du circuit en leur présence.

Mathew et Saab [81] ont étudié de leur côté l'effet du *reset* partiel sur la génération algorithmique et sur la couverture de pannes. En effet, les entrées *set/reset* sont des entrées additionnelles qui nous permettent d'avoir un contrôle plus grand sur l'état de la machine. Une utilisation appropriée de ces entrées peut résulter en un gain de testabilité, en plus de leur utilisation pour assurer l'initialisation. Cette idée a été exploitée par Abramovici et al. [2]. Ainsi, ils ont proposé une méthode de sélection des bascules pour assurer l'initialisation en se basant sur leurs contributions au test du circuit. Les bascules qui ont le plus d'effet sur la couverture de pannes et qui assurent l'initialisation sont retenues pour le *reset* partiel.

Les techniques discutées ci-haut sont essentiellement destinées au test algorithmique. Dans un contexte de test pseudo-aléatoire, d'autres problèmes d'initialisation peuvent surgir. Par exemple, Nadeau-Dostie et al. [94] ont soulevé le problème des bus dans une machine mal initialisée. Si plusieurs amplificateurs sont activés simultanément, des conflits peuvent être générés, et ainsi amener la machine en question dans des états invalides. Ceci entraînera éventuellement une corruption de la signature. Le *reset* complet peut être vu comme solution possible au problème des bus, cependant, il ne s'agit que d'une solution partielle et coûteuse. Ceci est un problème fondamental du test pseudo-aléatoire dans le cas des circuits avec bus. Des solutions pour ce problème ont déjà été proposées [94]. Une des solutions consiste à concevoir la machine de manière à ce que la logique contrôlant les bus ne puisse jamais activer plus d'un amplificateur par état. Dans le présent chapitre, nous supposons que tous les circuits sont conçus selon la solution mentionnée précédemment.

Ben Hamida et Kaminska [17][18] ont récemment proposé une mesure pour quantifier l'initialisation dite l'initiabilité. L'initiabilité vient s'ajouter aux notions de contrôlabilités classiques pour former un ensemble de mesures pour les circuits séquentiels. L'initiabilité d'un noeud séquentiel est définie comme étant la probabilité de le contrôler au temps "t+1", sachant qu'il ne l'était pas au temps "t". Une mise en oeuvre d'un programme pour calculer les initiabilités, appelée TOSTA, a également été proposée par les mêmes auteurs. Les expériences rapportées dans [17][18] ont montré que les initiabilités peuvent osciller en fonction du temps "t". La convergence vers une limite stable n'est parfois pas garantie. Dans ces conditions, nous ne pouvons pas affirmer avec certitude que l'initialisation est atteinte. Dans le présent chapitre, nous proposerons une généralisation des mesures COP classiques pour inclure les circuits séquentiels. Ces mesures, comme nous le verrons, quantifient l'initialisation et le test sans avoir besoin de dédier une mesure

spécifiquement à l'initialisation. De plus, nous démontrerons que ces mesures sont non-décroissantes en fonction du temps.

2.2 Coût du *reset* complet

Dans cette section, nous évaluerons l'impact d'un *reset* complet sur la surface d'un circuit. Pour cette évaluation, nous considérons plusieurs mises en oeuvre de bascules et de flip-flops de type D, ainsi que leurs équivalents avec entrées *set/reset*. La méthode de calcul de la circuiterie ajoutée (après insertion d'une bascule avec entrée *set/reset*), présentée ici, est inspirée de la méthode proposée par DasGupta et al. [42], méthode qui fut largement utilisée par plusieurs auteurs [4][98]. Dans ce type d'analyse, l'ajout de circuiterie est souvent exprimé en fonction de K, défini comme étant le rapport entre le nombre de portes dans la partie combinatoire du circuit et le nombre de bascules ou de flip-flops existant dans le même circuit (voire appendice 1). Selon [42], K varie entre 5 à 25 dépendamment du genre d'application visée avec le circuit.

Considérons la bascule de la figure 2.3 et son équivalent avec entrée *reset* conçue au niveau porte. La circuiterie ajoutée pour introduire le mecanisme de mise-à-0 coûte 1 porte par bascule. Par conséquent, la circuiterie ajoutée due à un *reset* complet peut s'écrire comme suit:

$$O_1 = \left(\frac{1}{K+5}\right) \times 100\% \qquad (2.1)$$

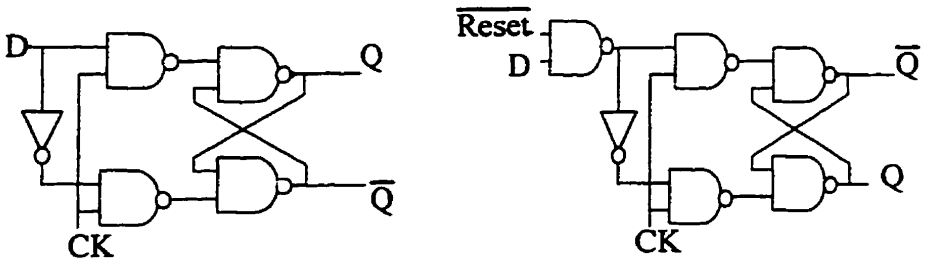


Figure 2.3 Une mise en oeuvre d'une bascule de type D et de son équivalent avec entrée de mise à 1

La figure 2.4 montre l'évolution de O_1 en fonction de K. Nous constatons ici que la circuiterie ajoutée peut atteindre 10% de la surface totale du circuit lorsque $K = 5$, et elle est plus de 5% pour $K < 15$.

Un autre cas intéressant est le flip-flop de type D avec son équivalent, conçus tous les deux au niveau transistor (voir figure 2.5). Ces circuits sont inspirés de la bibliothèque de cellules normalisées CMOS4S de la société canadienne de micro-électronique [29]. La version avec entrée *set* contient 20 transistors au total, 4 transistors de plus que la version simple (sans entrée *set/reset*), par conséquent, nous pouvons exprimer la circuiterie ajoutée due à un set/*reset* complet comme suit:

$$O_2 = \left(\frac{4}{4K+20}\right) \times 100\% = \left(\frac{1}{K+5}\right) \times 100\% \qquad (2.2)$$

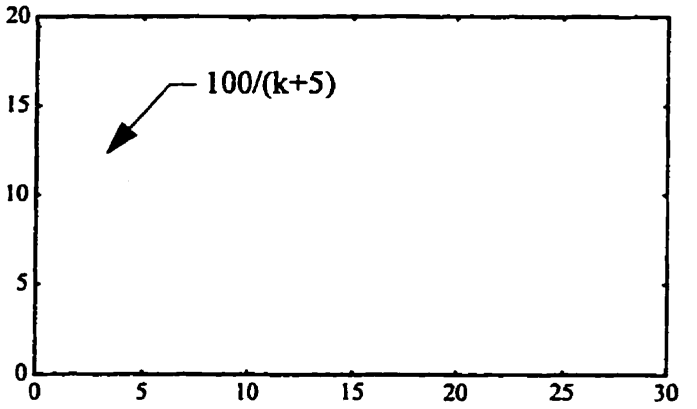


Figure 2.4 L'évolution de la circuiterie ajoutée en fonction du rapport K

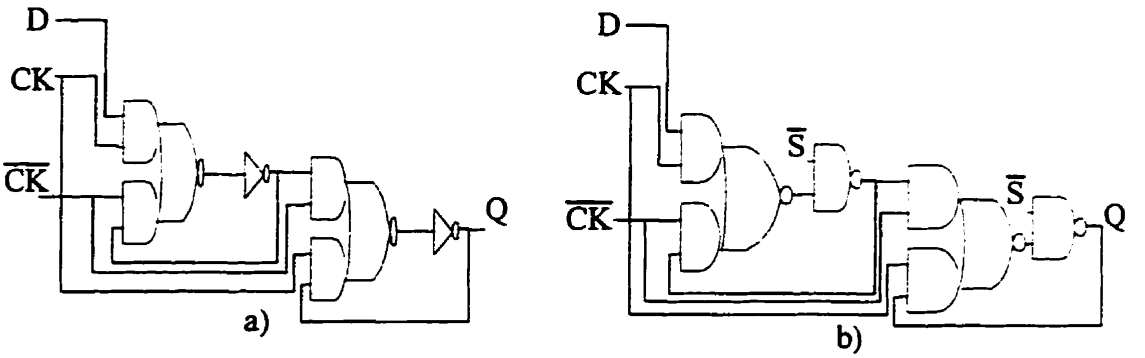


Figure 2.5 Une mise en oeuvre d'une flip-flop de type-D et son équivalent avec mécanisme de mise-à-1

Nous constatons ici que l'équation 2.2 est identique à l'équation 2.1. Notons que le nombre de flip-flops nécessaires est généralement plus petit que le nombre de bascules pour réaliser une fonctionnalité donnée. Dans certains cas, il peut être jusqu'à deux fois moindre. Néanmoins, même dans un circuit conçu avec des flip-flops comme ceux de la figure 2.5, la circuiterie ajoutée pour introduire un *reset* complet peut atteindre des proportions importantes.

La méthode de calcul de la circuiterie ajoutée adoptée ici n'est évidemment qu'approximative. Des méthodes plus élaborées tenant compte de la réalisation physique nous donneraient certainement des estimations plus précises. Malheureusement, à notre connaissance, aucune caractérisation détaillée de la circuiterie ajoutée au niveau physique n'est publiée dans la littérature. Quelques chiffres empiriques sont parfois publiés à titre indicatif seulement. Par exemple, Abramovici et al. [2] estiment que les bascules avec mécanismes de mise-à-0 / mise-à-1 sont à peu près 10% plus petites que les bascules avec des mécanismes de balayage. Dans [8], Agrawal et al. ont mentionné que le balayage complet peut introduire des ajouts de circuiterie qui peuvent atteindre 30% dans certains cas, dépendamment du type du balayage utilisé. Si ces deux chiffres sont représentatifs de la réalité, nous pouvons conclure que le *reset* complet peut introduire des circuiteries ajoutées qui sont supérieurs à 17% de la surface totale du circuit (voir appendice 1). Le tableau 2.1 donne des dimensions au niveau physique de plusieurs bascules tirées de différentes bibliothèques de cellules standards. Ces dimensions sont normalisées par rapport à la taille d'une bascule simple dans la même technologie et réalisée selon la même approche. Par exemple, la taille de la bascule S-R dff est 25% plus grande que son équivalent simple.

Tableau 2.1 Aires relatives de plusieurs bascules (dff) ainsi que leurs équivalents avec mise-à-0 (R-dff), mise-à-0/1 (S-R dff) et avec mécanisme de balayage (dffscan).

Libraries	dff	R-dff	S-R dff	dffscan
CMOS3DLM[30]	1	-	1.18	-
CMOS4S [29]	1	-	1.33	-
CMOSN [87]	1	1.12	1.25	1.5
LCA300K [78]	1	1.14	1.28	1.28
HCMOS [79]	1	1.2	1.4	1.6

Notons ici qu'en plus de la circuiterie ajoutée, un *reset* complet peut altérer les performances du circuit. Ceci est particulièrement vrai dans le cas des applications à hautes per-

formances. Dans le cas des bascules de la figure 2.3 et de la figure 2.5, nous pouvons déjà constater que les portes ajoutées pour forcer une mise-à-1 sont sur les chemins critiques de ces bascules.

2.3 Résumé de l'article de la section 2.4

Dans l'article qui fera l'objet de la section 2.4, paru dans IEEE Transactions on Computers en octobre 1995, vol. C-44, No. 10, pp. 1251-1256, nous étudierons le problème de l'initialisation des circuits séquentiels dans un contexte de test pseudo-aléatoire. Comme mentionné précédemment, l'initialisation dans un contexte de test pseudo-aléatoire est généralement accomplie par l'application d'une séquence de vecteurs déterministes ou bien par un matériel spécifique dédié tel que le *reset* complet.

Dans cet article, nous montrerons que l'initialisation avec des vecteurs pseudo-aléatoires est faisable. Nous proposons un modèle basé sur des chaînes de Markov modifiées pour modéliser les circuits séquentiels. Ce modèle nous permettra de montrer que l'initialisation en fonction du nombre de vecteurs appliqués est une fonction non décroissante. Cette propriété est à la base du concept d'initialisation par des vecteurs pseudo-aléatoires. En effet, la non-décroissance de la fonction d'initialisation nous montre qu'il est exclu de retourner à un état inconnu (non initialisé) à partir d'un état connu. C'est là, la principale contribution de cet article.

À partir du modèle markovien proposé, nous avons également développé un ensemble de règles de calcul des mesures de testabilité similaires à celles développées dans [17]. Une méthode de calcul itérative nommée sCOP (de sequential COP) pour propager ces mesures inspirée de [17], fut aussi développée. Notre méthode est différente de la méthode proposée dans [17], car sCOP calcule les mesures itérativement, sans avoir à couper artificiellement les lignes de retour.

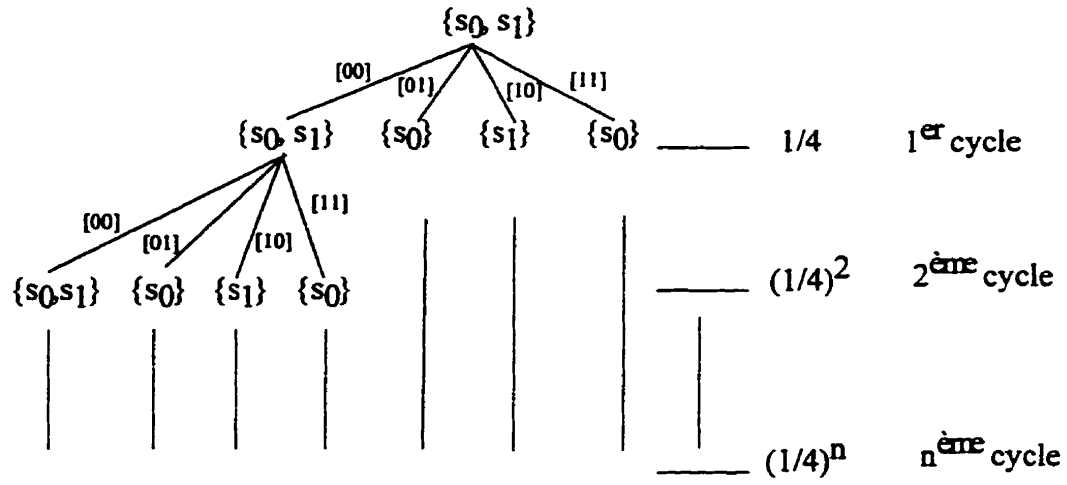


Figure 2.6 L'arbre de synchronisation pour "n" cycles de l'exemple montré à la Figure 2.7

Pour comprendre le principe de l'initialisation par des vecteurs pseudo-aléatoires, considérons l'exemple de la figure 2.7 de la page 39. La figure 2.6 représente l'arbre de synchronisation de ce circuit. L'application du vecteur "[01]", par exemple, fait transiter la machine d'un état complètement inconnu $\{s_0, s_1\}$ vers l'état connu $\{s_0\}$. En général, si nous assumons que les vecteurs d'entrées sont équiprobables et que l'état initial de la machine est complètement inconnu, la probabilité que cette machine soit encore non initialisée après un cycle est

$$\text{Probabilité de non initialisation} = \frac{1}{4}$$

Après 2 cycles,

$$\text{Probabilité de non initialisation} = \left(\frac{1}{4}\right)^2$$

et après "n" cycles,

$$\text{Probabilité de non initialisation} = \left(\frac{1}{4}\right)^n$$

Par conséquent, la probabilité d'initialisation de ce circuit après "n" cycles est

$$\text{Probabilité d'initialisation} = 1 - \left(\frac{1}{4}\right)^n$$

Nous pouvons déjà constater à partir de cet exemple que la probabilité d'initialisation est une fonction croissante de "n" et qu'elle converge vers 1 après quelques itérations seulement.

Des résultats expérimentaux nous ont confirmé que l'initialisation avec séquences de vecteurs de test pseudo-aléatoires est faisable. Nous avons conduit plusieurs expériences sur un grand ensemble de circuits d'essais séquentiels. Sur les 30 circuits considérés, 20 circuits ont été déclarés faciles à initialiser. Des simulations logiques (en utilisant Verilog-XL de Cadence) furent également invoquées pour confirmer l'initialisation.

2.4 Producing Reliable Initialization and Test of Sequential Circuits with Pseudo-Random Vectors .

M. Soufi, Y. Savaria, F. Darlay and B. Kaminska

Department of Electrical and Computer Engineering

Ecole Polytechnique de Montréal

P.O. Box 6079, Station "Centre-Ville", Montréal, Canada (H3C 3A7)

ABSTRACT

In this paper, the initialization of sequential circuits using ***pseudo-random*** *input patterns is addressed. An extended Markov chain model that covers the initialization phase is proposed. This model supports the theoretical framework used to demonstrate that sequential circuits can be initialized with* ***pseudo-random*** *vectors. This leads to a uniform BIST approach in which initialization and testing are performed together with a single pseudo-random generator.*

2.4.1 Introduction

Sequential circuit testing usually requires a prior initialization. Indeed, if the circuit cannot be fully initialized, some faults may remain untestable, and a single uninitialized bit is enough to corrupt the expected good signature with any signature analysis scheme. This initialization may be complex, and could require up to 10000 vectors for complex circuits [22]. Moreover, even with PR testing, a deterministic initialization is often essential, and if it is obtained by a deterministic sequence, some additional hardware devoted to the application of such a sequence may still be needed.

In order to guarantee initializability, set/reset inputs can be required for every flip-flop, in which case one pattern can determine the complete system state [72]. However, it may be unnecessary to force a deterministic initialization on flip-flops that are used as delay elements or for retiming, for example. With many designs, especially board designs using SSI and MSI components, initialization can be accomplished by using the asynchronous preset/clear inputs to feed every flip-flop [4]. In this paper, we are specifically interested in the initialization of sequential circuits with PR vectors. In fact, we propose a design for PR initialization. We will show how sequential circuits can be reliably initialized with PR vectors. Our work supports the use of a PR testing approach, in which initialization and test are performed together with a single PR generator.

This paper is organized as follows. In section 2.4.2, we propose a modified Markov chain model that accurately models the initialization phase of sequential circuits. In section 2.4.3, the initialization probability of sequential circuits is addressed. Some results are presented in section 2.4.4 and our main conclusions are summarized in section 2.4.5.

2.4.2 Initialization of Sequential Circuits in Pseudo-Random Testing

As mentioned earlier, testing a sequential circuit generally requires a prior initialization to put the circuit in a known state. This state should be uniquely determined for deterministic testing. However, in PR testing, if it is necessary to know precisely in which state the circuit has been initialized, it is not essential to control it precisely. Indeed, if the circuit is in one of its recurrent states as discussed below, its responses can be determined provided that PR input patterns are reproducible.

2.4.2.1 Modified Markov Chain Model

Lin et al. [75] and Youssef [146] proposed generalized COP measures for sequential circuits, where loops are modeled as Markov chain stochastic processes. According to these authors, the state transition diagram of a sequential circuit can be modeled with a Markov process, as shown in Figure 2.7-b.

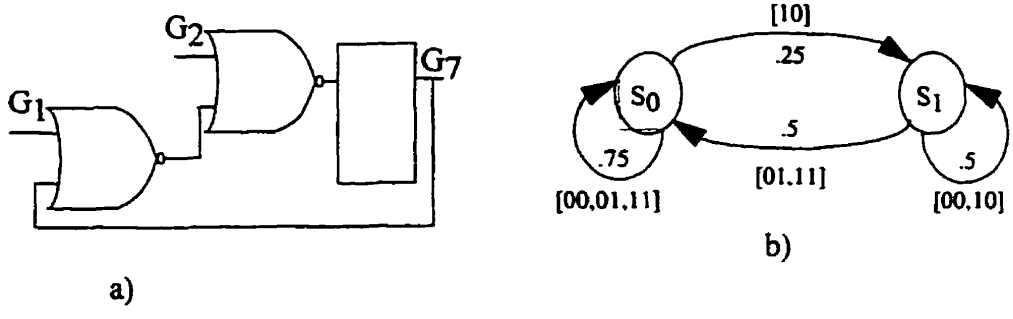


Figure 2.7 a) A sequential circuit taken from the ISCAS s27 circuit, b) its state transition diagram and the corresponding Markov chain diagram as defined by Lin et al. and Youssef

This chain can be completely described by means of a matrix (M) called the probability transition matrix. This matrix reflects the probabilities of taking specific transitions in one step between the circuit states. For the circuit in Figure 2.7, this matrix is:

$$M = \begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix} = \begin{bmatrix} 0{,}75 & 0{,}25 \\ 0{,}5 & 0{,}5 \end{bmatrix}$$

where s_{ij} is the transition probability, between state i and state j, in one step.

At each state, we can compute the probability of being in either state in Figure 2.7-b ($\pi_{s_0}(i,t)$ and $\pi_{s_1}(i,t)$), given that the circuit was initially in a known state. For instance, given that the initial state is "i", the probability vector describing the states at time "t" is

$$\Pi(i,t) = \Pi(i,0) \times M^t \tag{2.3}$$

where $\Pi(i,t) = \left(\pi_{s_0}(i,t), \pi_{s_1}(i,t)\right)$.

In the following, we propose a modified Markov chain process to capture the initialization phase. The corresponding extended Markov chain diagram for the previous sequential loop is depicted in Figure 2.8. In our model, we add a third possible state "X" (meaning unknown) to each node, and the initial state is the one where all circuits nodes are at "X".

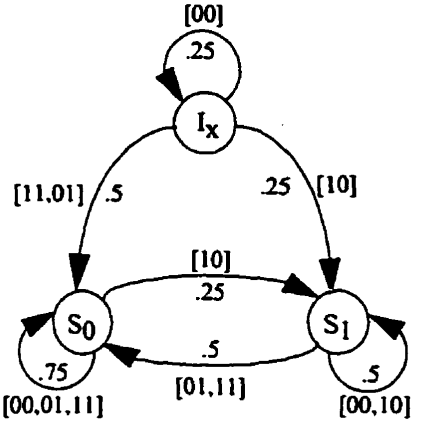


Figure 2.8 The state transition diagram of the extended Markov chain model of the example in Figure 2.7

This model automatically takes into account the initialization phase of sequential circuits. Initially, we start from the "unknown" state I_X, and at each time step "t", the probability $\pi_j(t)$ [1] of being in state "j" can be computed by means of equation 2.3 (where the starting state "i" of equation 1 is "I_X" in this case), which applies to all Markov chains.

1. Since the initial state is, by assumption, always I_X. This will not be explicitly specified in the rest of the paper, and thus $\pi_j(I_x,t) = \pi_j(t)$.

The probabilities $\pi_j(t)$ as a function of the time steps "t" for the above loop are shown in Figure 2.9. Note that the probability of being in state I_X decreases to zero. However, $\pi_{s_{11}}(t)$ and $\pi_{s_1}(t)$ increase, but they stabilize after a few time steps. Note also that the initialization probability defined as $P_{in}(t) = \pi_{s_0}(t) + \pi_{s_1}(t)$ is as high as ".99" at $t = 4$. This means that among all possible PR sequences of length l=4, 99% do initialize the circuit in a predictable and reproducible manner. This interpretation of the initialization probability gives us a measure of the probability that a single logic simulation of an initialization sequence can determine whether or not a circuit has been initialized. Note that, if the signature of the good circuit is to be determined by simulation, at least one simulation must always be performed.

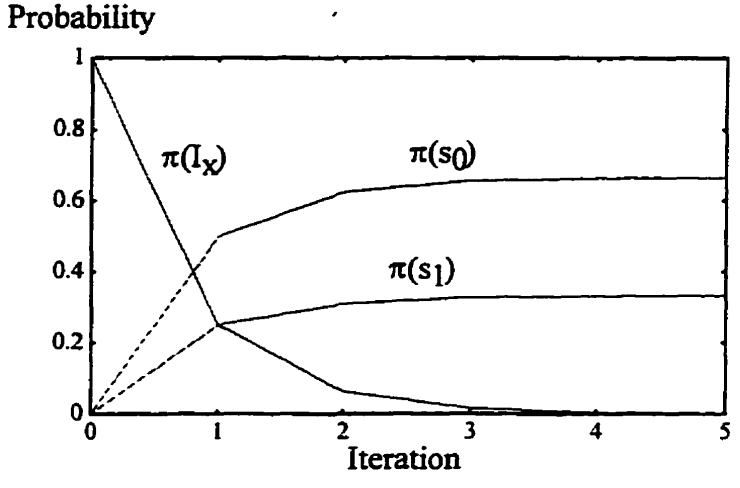


Figure 2.9 Curves representing the probabilities $\pi(s_0)$, $\pi(s_1)$ and $\pi(I_X)$ as functions of time "t"

Generally, the state diagram of a digital circuit is composed of two subsets, as depicted in Figure 2.10: a transient subset T, which includes all states where at least one flip-flop is at "X", and a recurrent subset R, which includes all states where no flip-flop is at "X". A fully initializable circuit should drift to R after a finite number of time steps, and

it should stay there unless there are illegal assignments generating unknown states. This is the basis of sequential circuit initialization with sequences of PR vectors [41].

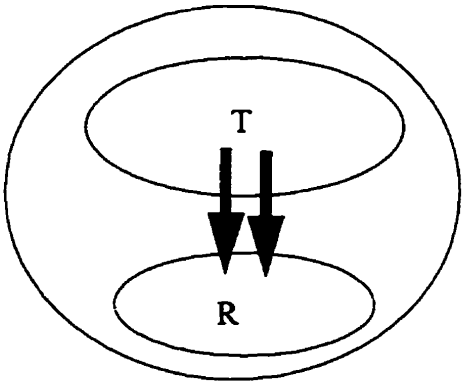


Figure 2.10 Markov chain of a digital circuit.

2.4.2.2 Testability Measures

From the general model depicted in Figure 2.10, and using the definitions provided in sections 2.4.2.1, we define the following controllabilities:

The controllability at 1 of a flip-flop "h" at time "t":

$$C_1(h,t) = \sum_{j\,(\text{where } h=1)} \pi_j(t) \tag{2.4}$$

The controllability at 0 of a flip-flop "h" at time "t":

$$C_0(h,t) = \sum_{j\,(\text{where } h=0)} \pi_j(t) \tag{2.5}$$

The controllability at "X" of a flip-flop "h" at time "t":

$$C_x(h,t) = \sum_{j\,(\text{where } h=X)} \pi_j(t) \tag{2.6}$$

$$\text{with } C_1(h,t) + C_0(h,t) + C_x(h,t) = 1 \tag{2.7}$$

Unfortunately, it is not practical to compute controllabilities from the previous equations. Indeed, as circuit complexity increases, the size of the transition matrices

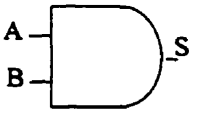
grows as 3^N (where "N" is the total number of flip-flops). Below, we will show that it is possible to directly compute approximations of these values, called sCOP numbers, from the circuit structures, in a manner similar to the one proposed in TOSTA [17].

2.4.2.3 sCOP algorithm

The proposed formulas to approximate $C_1(h,t)$, $C_0(h,t)$ and $C_x(h,t)$ are obtained from enlarged truth tables for each gate taking into account that each internal node of the circuit can be at "X". In the following example, we show how the different controllabilities for an AND gate are computed. This can easily be generalized to all gates (refer to [128]).

Considering the AND gate and its enlarged truth table of Figure 2.11, we have

$$\text{prob}\{S \neq 0 \text{ at time t}\} = \text{prob}\{(A \neq 0 \text{ at time t}) \text{ and } (B \neq 0 \text{ at time t})\}$$



A	B	S
0	0	0
0	1	0
1	0	0
1	1	1
0	X	0
1	X	X
X	0	0
X	1	X
X	X	X

Figure 2.11 Enlarged truth table for an AND gate.

Therefore, if we assume that controllabilities to zero of A and B are independent, then,

$$1 - C_0(S,t) = \left(\left(1 - C_0(A,t)\right) \times \left(1 - C_0(B,t)\right)\right)$$

and thus,

$$C_0(S,t) = 1 - \left(1 - C_0(A,t)\right) \times \left(1 - C_0(B,t)\right).$$

Since,

$$C_1(S,t) = \text{Prob}\{A = 1 \quad \text{and} \quad B = 1 \quad \text{at time t}\},$$

therefore,

$$C_1(S,t) = C_1(A,t) \times C_1(B,t),$$

and obviously we have, $C_0(S,t) + C_1(S,t) + C_x(S,t) = 1$.

The computation of sCOP values requires the following steps:

1. Set the initial controllabilities $C_1(h,0)$, $C_0(h,0)$ to zero for each internal node;
2. For each primary input I, set $C_1(I,t)$, $C_0(I,t)$ to .5;
3. Following a ranked order of circuit nodes "h", for "t" varying from 0 to the required time, compute $C_1(h,t)$ and $C_0(h,t)$ for all "h", from $C_1(h,t-1)$ and $C_0(h,t-1)$.

2.4.3 Initialization Probability of Sequential Circuits

Below, we show that the initialization probability $P(t)$ at each step "t" is equal to the sum of the probabilities of all the recurrent states. We also show that this initialization probability is a non-decreasing function of time t. Since the initialization probability is hard to compute, we will define two easily computed bounds for $P_x(t) = 1 - P(t)$ to declare whether or not a sequential circuit is initializable. Moreover, if we assume that the flip-flops are independent, we can compute an approximation for the overall initialization probability.

Theorem 1: The probability that a circuit C is initialized with a PR sequence of length "t" is given by

$$P(t) = \sum_{j \in R} \pi_j(t) \qquad \text{where R is the set of all recurrent states}$$

Proof: It is equivalent to say that a "circuit is initialized with a PR sequence of length t" and that "at time t, the circuit "C" is in one state of the subset R" (see Figure 2.10). This follows from the fact that once a circuit is fully initialized, no new "X" state can appear. Moreover, at some time "t", the events "be in state i" and "be in state j" are disjoint. Thus, their associated probabilities may always be summed when describing their union, and

$$P(t) = \sum_{j \in R} \pi_j(t) \tag{2.8}$$

Q.E.D

Theorem 2: $P(t)$ is a non-decreasing function of the time step "t".

Proof: Instead of providing a general proof of this theorem, we only show that it holds in a simple case: a general one flip-flop circuit. This simplified proof can be directly generalized to circuits with more than one flip-flop (refer to Appendix 3 for a sketch of the proof for the general case).

Without loss of generality, the transition matrix of the circuit depicted in Figure 2.12 is:

$$M = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & g \end{bmatrix}$$

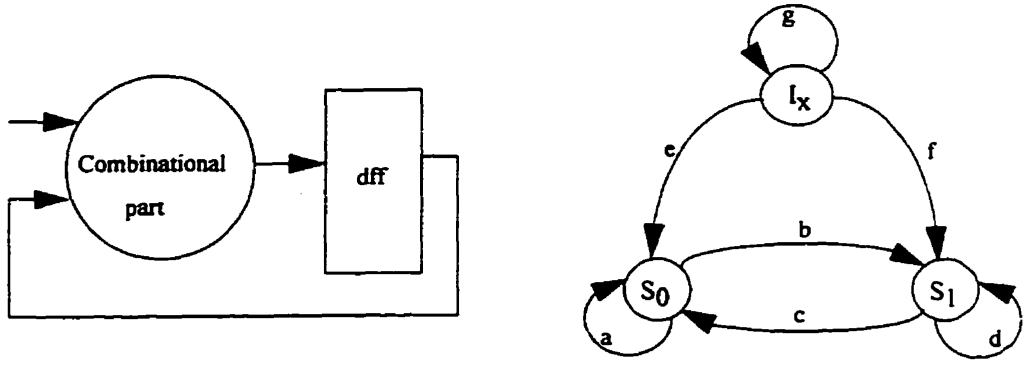


Figure 2.12 Example of a circuit with three states

If we express the probability vector quantifying the probability of being at time "t" in one of the states as $\Pi(t) = (\alpha,\beta,\gamma)$, where I_X is the initial state

then,

$$\begin{aligned}\Pi(t+1) &= \Pi(0) \times M^{(t+1)} \\ &= \Pi(0) \times M^{t} \times M \\ &= \Pi(t) \times M \\ &= (\alpha a + \beta c + \gamma e, \alpha b + \beta d + \gamma f, \gamma g)\end{aligned}$$

As mentioned in theorem 1, the initialization probability at time "t+1" is given by

$$\begin{aligned}P(t+1) &= \pi_{s_0}(t+1) + \pi_{s_1}(t+1) \\ &= \alpha a + \beta c + \gamma e + \alpha b + \beta d + \gamma f \\ &= \alpha(a+b) + \beta(c+d) + \gamma(e+f)\end{aligned}$$

Using the fact that the summation of all transition probabilities associated with any state is always 1, we have:

$$\begin{aligned}P(t+1) &= \alpha + \beta + \gamma(e+f) \\ &= P(t) + \gamma(e+f) \\ &> P(t) \qquad \text{if } \gamma(e+f) \neq 0\end{aligned}$$

Thus, in circuits with one state-variable, where known inputs do not generate unknown states, if the circuit is initializable, $P(t)$ is a strictly increasing function of "t",

and the only exception is when the circuit cannot be initialized, in which case $P(t) = 0$ for all "t". Q.E.D

In general, the exact computation of $P(t)$ has excessive computational complexity due to the 3^N dimensionality of the matrices. However, it is possible to approximate the probability of being in a state with at least one "X", $P_x(t) = 1 - P(t)$, by two easily computed bounds, as shown below (Theorem 3). One should note here that the exact computation of these bounds using Markov chain processes is also a time consuming process. However, these bounds may be directly approximated using sCOP numbers, which have a linear computational complexity.

Theorem 3:

$$L \le P_x(t) \le U \tag{2.9}$$

with

$$L = max\left(C_x\left(h_i, t\right)\right) \tag{2.10}$$

and

$$U = \sum_{i=1}^{..} \left(C_x\left(h_i, t\right)\right) \tag{2.11}$$

where i = 1,2..... N and "N" is the total number of flip-flops in the circuit.

Proof: From previous definitions, we can express $P_x(t) = 1 - P(t)$ as the sum of the individual state probabilities of all states in the subset of transient states T. Thus,

$$P_x(t) = \sum_{j \in T} \pi_j(t) .$$

Moreover, we have already defined $C_x(h_i, t)$ as (equation 2.6)

$$C_x(h_i, t) = \sum_{j \in T \text{ and } h_i = X} \pi_j(t),$$

thus $C_x(h_i, t)$ is a sum of a subset of the transient states, where $h_i = X$. However, $P_x(t)$ is the probability of the union of all the transient states. Therefore

$$C_x(h_i, t) \leq P_x(t) \qquad \text{for every } h_i.$$

Thus the first part of the inequality is

$$max(C_x(h_i, t)) \leq P_x(t)$$

The second part of the inequality can be developed as follows:

$$\sum_{i=1}^{..} C_x(h_i, t) = C_x(h_1, t) + C_x(h_2, t) + \ldots + C_x(h_N, t).$$

This sum includes all the transient states. However, since they are not disjoint, the probabilities associated with some states are counted more than once. Therefore,

$$P_x(t) \leq \sum_{i=1}^{..} C_x(h_i, t) \qquad \text{Q.E.D.}$$

Finally, it is also possible to compute an approximation for the initialization probability. The proposed expression is an approximation partly for the same basic reason that COP [53][54] and sCOP numbers, our proposed generalization to sequential circuits, are approximations. Indeed, all flip-flop groups controlling a node through a given gate are considered to be independent. Exploiting this assumption for the computation of the initialization probability, we can easily derive the following approximation of the initialization probability, called AIP(t)

$$AIP(t) = \prod_{i=1}^{..} P(h_i, t) \tag{2.12}$$

where $P(h_i, t)$ is the initialization probability of flip-flop "i" at time "t" defined as

$$P(h_i, t) = C_0(h_i, t) + C_1(h_i, t)$$

Recall that the bounds "U", "L" and the approximate initialization probability $AIP(t)$ are computed using sCOP, which is linear with respect to the number of gates in the circuit. Therefore, these parameters are only approximations. However, the results conducted on a large subset of the ISCAS'89 sequential benchmarks shows that they are in general sufficient.

2.4.4 Experimental Results

In the first experiment, the sCOP computation method was run on a large set of ISCAS 89 sequential benchmark circuits [26]. We computed the approximate initialization probability $AIP = 1 - \overline{AIP}$ with (equation 2.12), the lower and upper bounds of the complement of the initialization probability (U and L (equation 2.11), (equation 2.10), until they stabilized or until the number of iterations exceeded 2×10^4. The "AIP" and the bounds are declared to be "stabilized" when their values computed to an accuracy arbitrarily set at 10^{-16} do not change until the limit of 2×10^4 time steps is reached. The results are shown in Table 1, where a flip-flop is declared impossible to initialize when its

initialization probability $P(h_i, t)$ sticks at 0 indefinitely, and it is declared difficult to initialize if its initialization probability remains lower than a limit set arbitrarily to 0.99.

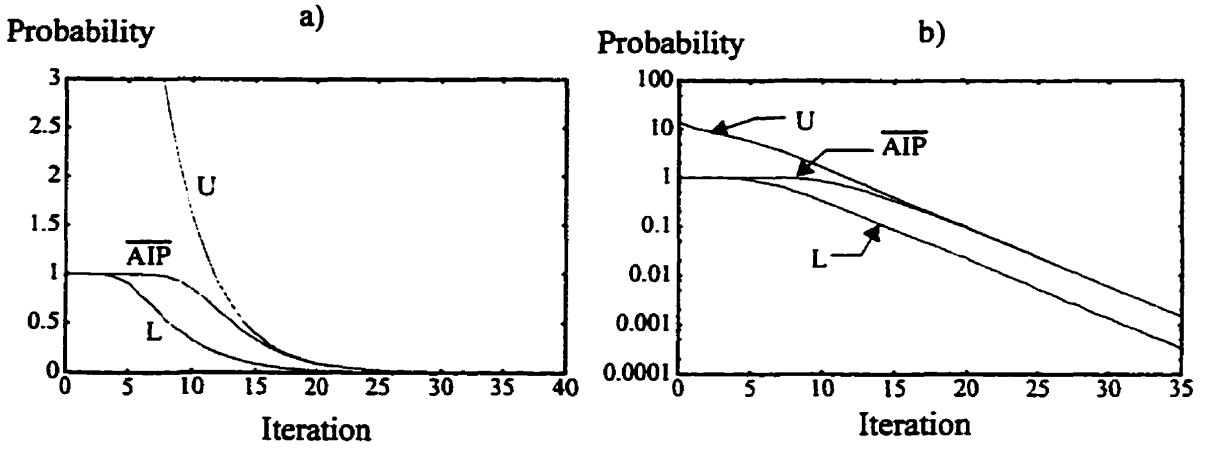


Figure 2.13 "U", "L" and "AIP" were non-increasing functions of the time "t" in all experiments as shown here for s298

Table 2.2 Computation results of "U", "L" and "AIP" until they stabilize or the number of iterations exceeds 20,000 vectors

Circuit name	# DFF	Upper bound		AIP		Lower bound		Impossible to initialize FFs		Hard to initialize FFs	
		U	V_u	AIP	V_{AIP}	L	V_l	#imp	V_{imp}	#hard	V_{hard}
s27	3	$< 10^{-16}$	54	$< 10^{-16}$	54	$< 10^{-16}$	54	0	1	0	7
s208	8	10^{-16}	56	10^{-16}	56	10^{-16}	56	0	2	0	8
s298	14	12×10^{-16}	144	12×10^{-16}	144	2×10^{-16}	144	0	4	0	23
s344	15	30×10^{-16}	130	30×10^{-16}	130	3×10^{-16}	128	0	3	0	22
s349	15	19×10^{-16}	183	19×10^{-16}	183	3×10^{-16}	178	0	3	0	29
s382	21	13×10^{-16}	141	13×10^{-16}	141	13×10^{-16}	141	0	2	0	20
s386	6	10^{-16}	173	10^{-16}	173	10^{-16}	173	0	2	0	25
s420	16	10^{-16}	56	10^{-16}	56	10^{-16}	56	0	2	0	8
s444	21	14×10^{-16}	175	14×10^{-16}	175	2×10^{-16}	175	0	2	0	8
s510	6	6	∞	1	∞	1	∞	6	1	6	1
s526	21	3×10^{-16}	140	3×10^{-16}	140	10^{-16}	140	0	4	0	22
s526n	21	4×10^{-16}	140	4×10^{-16}	140	10^{-16}	140	0	4	0	21
s641	19	60×10^{-16}	212	60×10^{-16}	212	8×10^{-16}	210	0	2	0	28
s713	19	130×10^{-16}	362	130×10^{-16}	362	16×10^{-16}	350	0	2	0	48
s820	5	9×10^{-16}	327	9×10^{-16}	327	3×10^{-16}	327	0	1	0	43

Table 2.2 Computation results of "U", "L" and "AIP" until they stabilize or the number of iterations exceeds 20,000 vectors

Circuit name	# DFF	Upper bound		AIP		Lower bound		Impossible to initialize FFs		Hard to initialize FFs	
		U	V_u	AIP	V_{AIP}	L	V_l	#imp	V_{imp}	#hard	V_{hard}
s832	5	13×10^{-16}	315	13×10^{-16}	315	4×10^{-16}	315	0	1	0	42
s838	32	10^{-16}	56	10^{-16}	56	10^{-16}	56	0	2	0	8
s953	29	21.5	4	1	∞	1	∞	19	1	26	2
s1196	18	$< 10^{-16}$	3	$< 10^{-16}$	3	$< 10^{-16}$	1	0	1	0	3
s1238	18	$< 10^{-16}$	3	$< 10^{-16}$	3	$< 10^{-16}$	1	0	1	0	3
s1423	74	112×10^{-16}	193	112×10^{-16}	193	4×10^{-16}	190	0	3	0	31
s1488	6	2.74	64	.97	64	.48	64	0	1	6	1
s1494	6	2.75	64	.97	68	.48	66	0	1	6	1
s5378	179	$36,17 \times 10^{-14}$	6075	$36,17 \times 10^{-14}$	6075	$4,13 \times 10^{-14}$	6069	0	15	0	1046
s9234	228	173.7	1642	1	∞	1	∞	172	1	182	203
s13207	669	480.5	2×10^4	1	∞	1	∞	406	74	522	13618
s15850	597	329.8	510	1	∞.	1	∞	420	79	286	24
s35932	1728	140×10^{-16}	147	140×10^{-16}	147	2×10^{-16}	128	0	1	0	18
s38417	1636	513.3	2×10^4	1	∞	1	∞	212	3079	732	7798
s38584	1452	554.9	1023	1	∞	1	∞	49	13	1417	262

1. #DFF: Number of FFs in the circuit
2. U, V_u: value of the upper bound U and the corresponding iteration when the bound stabilizes
3. L, V_l: value of the upper bound U and the corresponding iteration when the bound stabilizes
4. $\overline{AIP}$, $V_{\overline{AIP}}$: the complement of the approximate initialization probability ($\overline{AIP}$ = 1 - AIP) and its corresponding stabilization iteration ($V_{\overline{AIP}}$).
5. #imp, V_{imp}: the number of impossible-to-initialize flip-flops and the corresponding iteration when #imp stabilizes.
6. #hard, V_{hard}: the number of hard-to-initialize flip-flops and the corresponding iteration when #imp stabilizes.

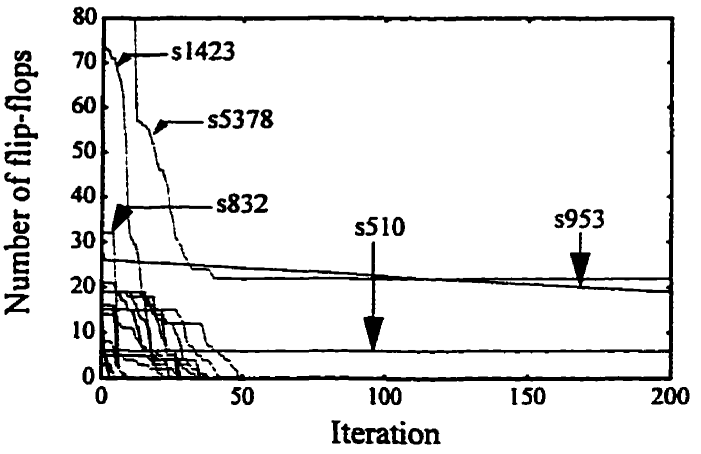


Figure 2.14 The number of flip-flops that are declared not initialized as a function of time step "t".

The results show that a majority of circuits have a good probability of being initialized after a reasonable number of PR vectors. The upper bound "U" of the circuit s713, for example, is as small as $\left(1{,}30 \times 10^{-14}\right)$ after 362 iterations. Therefore, the probability that this circuit will be initialized by a PR sequence of length 362 vectors is better than $(1 - 1{,}30 \times 10^{-14})$. This means that among all possible sequences of 362 vectors, $\left(100 - 1{,}30 \times 10^{-12}\right)\%$ do initialize the circuit in a predictable and reproducible manner, provided that the input vectors are PR and that the independence assumption which underlies sCOP calculations is a good approximation. Moreover, in all the experiments performed, the two bounds "U" and "L", as well as the "$\overline{\text{AIP}}$", were non-increasing functions of the time "t". Furthermore, for the circuits that were declared to be initialized, these functions were almost null after applying a small number of iterations, as shown in Figure 2.13-a for the case of s298. The same plots are depicted in Figure 2.13-b on a semilog scale to illustrate the fast convergence rates of each of the three parameters "U", "L" and "$\overline{\text{AIP}}$". However, some circuits (s510, s953, s1488,s1494,s9234, s13207,

s15850, s38417, s38584) cannot be easily initialized with a reasonable number of PR vectors, and they seem to be impossible to initialize without hardware modifications. Figure 2.14 shows the dynamic of the number of flip-flops, which are still not initialized, as a function of time "t". Note that a majority of flip-flops can be detected as initializable or not-initializable after a reasonable number of iterations. Therefore, it seems that a limit on the iteration number arbitrarily set to 100 may be sufficient to detect the majority of flip-flops which are not initialized.

2.4.5 Conclusion

To summarize, design for pseudo-random initialization was shown to be feasible. The proposed generalized testability measures, which are based on a modified Markov chain, were shown to be capable of predicting whether or not a sequential circuit is initializable. The complexity of computing these measures is linear with respect to the number of gates in the circuits

Results of experiments conducted on 30 sequential ISCAS 89 circuits were reported. Among these circuits, 20 were found to be easy to initialize with few pseudo-random vectors. This finding was not restricted to small circuits, and larger ones were also found to be easy to initialize. Moreover, we confirmed experimentally that the initialization probability is a non-decreasing function of the number of clock cycles, and that 100 clock cycles were in general sufficient to declare whether or not a sequential circuit is initializable.

2.5 Méthode de modification pour l'initialisation

Les résultats expérimentaux de la section 2.4 nous montrent qu'il existe une classe de circuits qui résistent à l'initialisation avec des vecteurs pseudo-aléatoires. Un exemple typique d'une telle classe est la bascule T. La bascule T, montrée sur la figure 2.15, est impossible à initialiser. À partir d'un état inconnu, nous ne pouvons jamais transiter à un état connu. Dans un contexte pseudo-aléatoire, il existe un autre type de problème d'initialisation où cette dernière, malgré qu'elle soit possible, exige de très longues séquences d'initialisation.

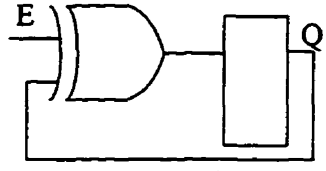


Figure 2.15 Bascule T qui est un exemple d'un circuit impossible à initialiser

Dans ce qui suit, nous proposons une méthode simple pour assurer l'initialisation des circuits séquentiels difficiles à initialiser avec des séquences d'initialisation de longueurs raisonnables. Avec cette méthode qui est basée sur les mesures de testabilité sCOP, un sous-ensemble de bascules est sélectionné comme candidats pour un *reset* partiel.

La méthode proposée ici consiste à définir un critère d'initialisation approprié pour la probabilité d'initialisation $p(h_i, t)$ de façon à ce qu'une bascule "h_i" soit déclarée initialisée si sa probabilité d'initialisation est supérieure au critère en question. Évidemment, nous pouvons toujours concevoir des contre-exemples. Le critère d'initialisation n'est qu'un critère heuristique que nous voulons efficace dans la majeure partie des cas. Nous proposons également dans cette méthode, un critère d'arrêt sur le nombre d'itérations. Ce critère est relié au fait que la complexité du calcul des nombres sCOP est fortement dépen-

dante du nombre d'itérations utilisées. En effet, le calcul des nombres sCOP est "n" fois plus complexe que celui des nombres COP, où "n" est le nombre d'itérations utilisées. De plus, dans le cas où une bascule est impossible à initialiser, nous risquons de faire tourner le programme sCOP indéfiniment sans atteindre le critère d'initialisation. Par conséquent, il est impératif de limiter le nombre d'itérations de sCOP. Avant d'aller plus loin, posons l'hypothèse heuristique suivante de laquelle découle le critère d'arrêt:

Hypothèse: Si une bascule est encore non initialisée après "S" itérations, elle ne sera pas initialisée avec "$S + \varepsilon$" itérations, où ε est raisonnablement petit et "S" est un critère d'arrêt qui sera déterminé à partir des résultats expérimentaux.

Nos résultats expérimentaux montrent que cette hypothèse est vérifiée avec une bonne probabilité. Encore une fois, l'hypothèse ci-dessus n'est qu'une hypothèse heuristique. Des contre-exemples peuvent être construits. L'argument de base sous-jacent de l'hypothèse est directement relié à un compromis entre le nombre d'itérations qui affecte principalement la complexité de calcul des nombres sCOP et la précision de prédire qu'une bascule ne puisse être initialisée. "S" dépend de la nature et de la complexité du circuit considéré.

2.5.1 Heuristique de mise-à-0 / mise-à-1

En utilisant cette hypothèse, nous proposons ici une méthode heuristique pour modifier les circuits difficiles à initialiser avec des vecteurs pseudo-aléatoires, dans le but de les rendre facile à initialiser. Avec cette méthode, toutes les bascules déclarées encore non initialisées après "S" itérations sont mises-à-0 / mises-à-1 en deux étapes. Dans la première étape, les bascules déclarées impossibles à initialiser sont modifiées. Dans la deuxième étape, le reste des bascules déclarées non initialisées est modifié. Une bascule est déclarée

impossible à initialiser si sa probabilité d'initialisation $p(h_i, t)$ reste collée à 0 jusqu'à la "$S^{ième}$" itération. Dans le cas où cette probabilité est différente de 0 mais inférieure à une limite (qui sera définie un peu plus bas), la bascule en question est dite difficile à initialiser.

La méthode de modification peut être résumée comme suit:

1. invoquer sCOP pour calculer $p(h_i, t)$ de toutes les bascules. Après "S" itérations, si $p(h_i, S) > 0{,}99$ pour toutes les bascules h_i, *exit*.
2. *else if* $p(h_i, S) = 0$ pour certains h_i,
   - •imposer une initialisation déterministe avec un poids de 0,5 sur toutes les bascules déclarées impossibles à initialiser;
   - •re-calculer les nombres sCOP pour toutes les bascules en commençant à partir de $t = 0$.
   - •si $p(h_i, S) > 0{,}99$ pour toutes les bascules h_i, *exit*.
3. *else if* $0 < p(h_i, S) < 0{,}99$ pour certains h_i,
   - •imposer une initialisation déterministe avec un poids de 0,5 sur toutes les bascules déclarées difficiles à initialiser;
   - •recalculer les nombres sCOP pour toutes les bascules en commençant à partir de $t = 0$, dans le but de vérifier si le circuit est effectivement devenu facile à initialiser.

Notons ici que le calcul des nombres sCOP est répété 3 fois, cependant, seulement "S" vecteurs pseudo-aléatoires, et non "$3 \times S$", qui sont appliqués pour garantir l'initialisation. Les modes utilisés dans la méthode de mise-à-0 / mise-à-1 sont des moyens pour évaluer la sensibilité des circuits aux changements, mais ils n'allongent pas la séquence d'initialisation.

2.5.2 Résultats expérimentaux

Le tableau 2.3 donne les résultats obtenus ralativement à l'initialisation avec et sans la procédure précédente d'insertion de points d'initialisation. Dans ce tableau, #dff se

rapporte au nombre de bascules ou un problème est découvert et V est le nombre d'itérations requises pour atteindre cette conclusion. Notez que quand V < 100, #dff est 0.

Les résultats montrent que le mode 1, où seulement les bascules déclarées impossibles à initialiser sont modifiées, est suffisant pour assurer l'initialisation de certains circuits (s510, s953, s15850). Cependant, le mode 1 s'avère être insuffisant pour d'autres circuits. Dans le cas du circuit s9234 par exemple, 2 bascules sont déclarées non initialisées après le mode 1 et ce n'est qu'après le mode 2 que le circuit est déclaré facile à initialiser. Notons que dans certaines situations, en l'occurrence le cas de s1488 et s1494, toutes les bascules déclarées non initialisées ne sont pas impossibles à initialiser. Dans ce cas, le mode 1 est sans utilité, nous pouvons ainsi passer directement au mode 2.

Tableau 2.3 Résultats de la méthode de mise-à-0 / mise-à-1

					mode 1				mode 2							
	Impossible à initialiser		Difficile à initialiser		Impossible à initialiser		Difficile à initialiser		Impossible à initialiser		Difficile à initialiser					
circuit	#dff	V	#dff	V	#dff	V	#dff	V	#dff	V	#dff	V	T	T%	A%	M%
s27	0	1	0	7	-	-	-	-	-	-	-	-	0	0	0	-
s208	0	2	0	8	-	-	-	-	-	-	-	-	0	0	0	-
s298	0	4	0	23	-	-	-	-	-	-	-	-	0	0	0	-
s344	0	3	0	22	-	-	-	-	-	-	-	-	0	0	0	-
s349	0	3	0	29	-	-	-	-	-	-	-	-	0	0	0	-
s382	0	2	0	20	-	-	-	-	-	-	-	-	0	0	0	-
s386	0	2	0	25	-	-	-	-	-	-	-	-	0	0	0	-
s420	0	2	0	8	-	-	-	-	-	-	-	-	0	0	0	-
s444	0	2	0	8	-	-	-	-	-	-	-	-	0	0	0	-
s510	6	100	6	100	0	1	0	1	-	-	-	-	6	100	1.11	0
s526	0	4	0	22	-	-	-	-	-	-	-	-	0	0	0	-
s526n	0	4	0	21	-	-	-	-	-	-	-	-	0	0	0	-
s641	0	2	0	28	-	-	-	-	-	-	-	-	0	0	0	-
s713	0	2	0	48	-	-	-	-	-	-	-	-	0	0	0	-
s820	0	1	0	43	-	-	-	-	-	-	-	-	0	0	0	-
s832	0	1	0	42	-	-	-	-	-	-	-	-	0	0	0	-
s838	0	2	0	8	-	-	-	-	-	-	-	-	0	0	0	-
s953	19	100	26	100	0	1	0	3	-	-	-	-	19	66	1.73	0
s1196	0	1	0	3	-	-	-	-	-	-	-	-	0	0	0	-
s1238	0	1	0	3	-	-	-	-	-	-	-	-	0	0	0	-
s1423	0	3	0	31	-	-	-	-	-	-	-	-	0	0	0	-
s1488	0	1	6	100	0	1	6	100	0	1	0	1	6	100	.39	100
s1494	0	1	6	100	0	1	6	100	0	1	0	1	6	100	.39	100
s5378	0	11	22	100	-	-	-	-	0	11	0	18	22	12	.35	100
s9234	172	100	206	100	0	2	2	100	0	1	0	85	174	76	1.67	18
s13207	406	100	540	100	0	20	72	100	0	10	0	49	478	71	2.88	54

Tableau 2.3 Résultats de la méthode de mise-à-0 / mise-à-1

					mode 1				mode 2							
	Impossible à initialiser		Difficile à initialiser		Impossible à initialiser		Difficile à initialiser		Impossible à initialiser		Difficile à initialiser					
circuit	#dff	V	#dff	V	#dff	V	#dff	V	#dff	V	#dff	V	T	T%	A%	M%
s15850	286	100	420	100	0	17	0	36	-	-	-	-	286	48	1.51	0
s35932	0	1	0	18	-	-	-	-	-	-	-	-	0	0	0	-
s38417	228	100	976	100	0	51	745	100	0	10	0	76	973	37	2.08	-99
s38584	49	100	1431	100	0	11	1058	100	0	8	0	81	1107	76	2.41	77

1. T: nombre total de bascules qui sont modifiées durant les deux modes.
2. %T: Pourcentage de T définie comme $\%T = \frac{T}{\text{nombre total de bascules dans un circuit (\#DFF)}}$
3. A%: circuiterie ajoutée
4. %M: Facteur de mérite du mode 1 définie comme $\%M = \frac{T - (\text{\#dff impossible à initialiser})}{(\text{\#dff non initialisees}) - (\text{\#dff impossible à initialiser})}$
5. IPI: Nombre de points d'initialisation insérés en utilisant TOSTA.
6. IPI%: Circuiterie ajoutée par les points d'initialisation de TOSTA.

Le pourcentage (%T) des bascules modifiées en utilisant cette méthode varie entre 0% et 100%. La moyenne de ce pourcentage, calculée pour tous les circuits considérés est de l'ordre de 38%. Nous avons également estimé le pourcentage de la circuiterie ajoutée en utilisant un modèle inspiré de l'équation 2.1. En effet, si nous supposons que seulement une fraction "n" des bascules a été modifiée, à partir de l'équation 2.1, nous pouvons écrire:

$$O_3 = \left(\frac{n}{K+5}\right) \times 100\% \tag{2.13}$$

Les résultats montrent que la circuiterie ajoutée varie entre 0% et 2.88% et sa moyenne sur tous les circuits considérés est de l'ordre de 0.44% seulement.

Le facteur de mérite "M%" est défini comme étant le pourcentage des bascules difficiles à initialiser modifiées en mode 2 par rapport au nombre total des bascules déclarées difficiles à initialiser. Ce facteur reflète l'efficacité du mode 1 à initialiser un circuit séquentiel. Par exemple, dans le cas de s9234, seulement 18% de toutes les bascules qui

pouvaient être potentiellement modifiées durant le mode 2 ont été effectivement modifiées durant ce mode. Par conséquent, 82% de ces bascules ont été rendues initialisables par le mode 1.

2.6 *Reset* partiel pour le test pseudo-aléatoire

Jusqu'à présent, nous avons utilisé le *reset* partiel pour assurer l'initialisation des circuits séquentiels. Dans ce qui suit, nous étudierons l'effet du *reset* partiel sur le test pseudo-aléatoire des circuits séquentiels. Nous montrerons ainsi que le *reset* partiel, judicieusement utilisé, améliore le test pseudo-aléatoire des circuits séquentiels dans certains cas. Cette section a fait l'objet d'un article publié à ISCAS 95 [126].

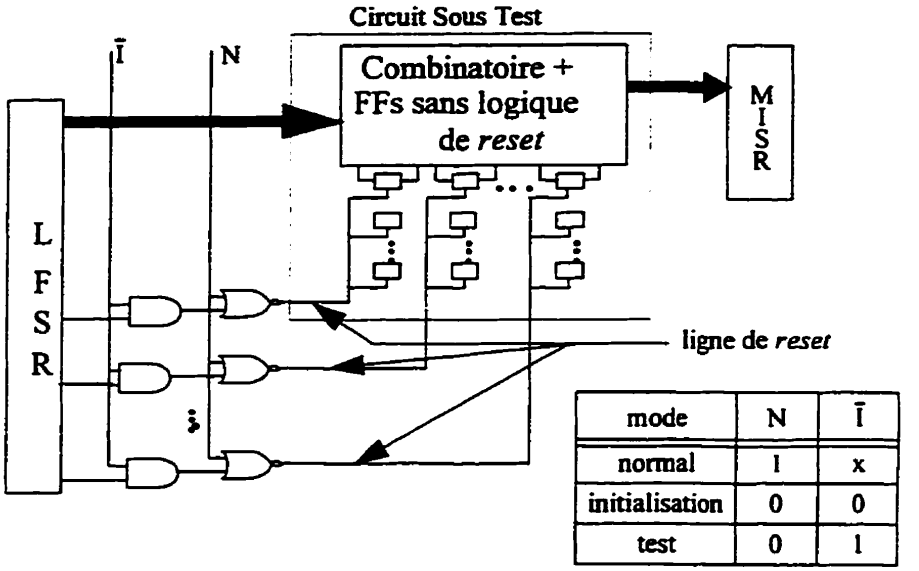
Le *reset* complet a la particularité d'amener le circuit à un état initial connu (état "*RESET*") à chaque fois que le *reset* est activé. Par conséquent, la ligne de *reset* doit être soigneusement utilisée durant le mode test, faute de quoi nous risquons de détruire l'état présent du système [2]. Ceci n'est pas le cas avec le *reset* partiel. L'activation du *reset* dans un circuit avec *reset* partiel nous amène à un état où les bascules sans entrée *reset* sont *a priori* dans un état "quelconque". Cet aspect non-déterministe du *reset* partiel peut contribuer à améliorer la qualité du test pseudo-aléatoire.

Cette idée fut exploitée par plusieurs auteurs [2][81][82][126]. Ainsi, dans [81], Mathew et Saab ont proposé un algorithme pour trouver un petit ensemble de bascules de manière à ce que toutes les lignes de retours soient coupées. Ces bascules sont modifiées pour former un *reset* partiel. Dans cette méthode, une seule ligne de *reset* est utilisée. Dans un autre article [82], les mêmes auteurs montrent que l'utilisation d'une seule ligne de *reset* est parfois contraignante. En effet, à chaque activation du *reset*, nous transitons vers un sous espace d'états déterminé par l'état des bascules avec entrée *reset/set*. Pour rendre le *reset* plus "aléatoire" et ainsi améliorer davantage la qualité du test, Mathew et al. [82]

ont proposé d'utiliser plusieurs lignes de *reset* pour alimenter les entrées *reset/set* des différentes bascules. Abramovici et al. [2] ont de leur coté proposé une technique de *reset* partiel basée sur la contribution des bascules à la testabilité. Pour cela, une analyse de sensibilité pour ordonner les bascules selon leur contribution à la qualité du test est utilisée. Les résultats obtenus par leur méthode semblent être systématiquement meilleurs que les autres techniques. Tous ces travaux sont destinés au test algorithmique. Dans la présente section, nous proposons une technique de *reset* partiel où la sélection des bascules est basée sur leurs contributions à la qualité du test pseudo-aléatoire. Cette technique nous permet ainsi d'assurer l'initialisation d'un côté et un bon niveau de qualité du test d'un autre côté.

2.6.1 Le système proposé

L'idée principale de la technique de conception pour le test proposée ici est illustrée dans la figure 2.16. Les bascules responsables des problèmes d'initialisation et celles responsables des problèmes de test sont remplacées par leurs équivalents avec des mécanismes de mise-à-0/mise-à-1. Notons que le balayage partiel n'est pas considéré ici, cependant, il peut être facilement ajouté si c'est nécessaire. Un générateur de vecteurs de test pseudo-aléatoires (LFSR par exemple) est utilisé pour alimenter les entrées primaires et les entrées *reset*. Durant le mode test ($\bar{I}$=1, N=0), les entrées *reset* sont alimentées par des sources pseudo-aléatoires provenant du générateur. En mode initialisation ($\bar{I}$=0, N=0), un *reset* est forcé sur toutes les bascules avec entrée *reset* et ceci durant toute la période d'initialisation. En même temps, les entrées primaires sont directement alimentées par des sources pseudo-aléatoires provenant du générateur.



mode	N	Ī
normal	1	x
initialisation	0	0
test	0	1

Figure 2.16 Structure descriptive du système BIST (Test intégré) proposé ici

Notons ici que le nombre d'étages additionnels requis, pour alimenter les entrées *reset*, peut être réduit par des techniques de condensation. En effet, il est possible d'alimenter les entrées *reset* par les entrées primaires ou bien à partir de n'importe quelle ligne du circuit, tant que l'hypothèse d'indépendance entre la bascule et la ligne source est respectée [126]. Finalement, un compacteur de signatures (généralement un MISR) est utilisé comme analyseur de signatures.

2.6.2 Heuristiques de sélection

La sélection optimale des bascules à remettre à zéro est un problème difficile. Dans ce qui suit, nous proposerons quelques heuristiques pour le résoudre. Une stratégie valide consiste à définir un critère approprié pour les contrôlabilités de manière à ce que chaque bascule soit remise à 0(1) si sa contrôlabilité à 1(0) estimée par les nombres sCOP est plus petite qu'un seuil fixé arbitrairement à 0,01 [129]. La figure 2.17 illustre les principales

étapes de cette stratégie. Après avoir appliqué l'heuristique d'insertion des points d'initialisation de la section 2.5.1, nous continuons le calcul des nombres sCOP pour tous les noeuds en alimentant les entrées *reset* des bascules retenues pour l'initialisation par des sources pseudo-aléatoires. Ensuite, une mise à zéro ou à un est appliquée sur toutes les bascules à controllabilité plus petite que le seuil selon la procédure montrée sur la figure 2.17.

Test1(circuit)

1. Modifier le circuit pour assurer l'initialisation (avec l'heuristique de la section 2.5.1)
2. Calculer les nombres sCOP pour tous les noeuds en alimentant les entrées *reset* des bascules retenues par l'étape 1 par des sources pseudo-aléatoires
3. Pour chaque bascule "h"
   if($C_0(h,t) < 0{,}01$) appliquer une mise à 0 sur la bascule "h".
   else if($C_1(h,t) < 0{,}01$) appliquer une mise à 1 sur la bascule "h".

Figure 2.17 Procédure 1: ébauche de l'heuristique 1.

Cette heuristique donne des résultats valables malgré sa simplicité. Cependant, il faut noter ici qu'elle peut être coûteuse en terme du nombre de bascules sélectionnées pour le *reset*. En effet, avec cette heuristique, toutes les bascules à faible contrôlabilité sont remise

à zero/un sans distinction entre celles qui sont responsables des problèmes de test et celles qui sont les conséquences d'un problème de test sur d'autres bascules.

Test2(circuit)

1. Modifier le circuit pour assurer l'initialisation (avec l'heuristique de la section 2.5.1)
2. Ordonner les bascules selon l'ordre de leurs profondeurs séquentielles
3. Pour chaque bascule "h"
4. Calculer les nombres sCOP pour tous les noeuds en alimentant les bascules déjà retenues pour le *reset* partiel (initialisation et test) par des sources pseudo-aléatoires
5. Calculer la couverture de pannes FC1 (en utilisant les nombres sCOP)
6. if($C_0(h,t) < 0{,}01$)
   appliquer une mise à 0 sur la bascule "h"
   calculer la nouvelle couverture de pannes FC2
   if(FC2 - FC1 < q) enlever la mise à 0 appliquée sur "h"
7. else if($C_1(h,t) < 0{,}01$)
   appliquer une mise à 1 sur "h"
   calculer la nouvelle couverture de pannes FC2
   if(FC2 - FC1 < q) enlever la mise à 1 appliquée sur "h"

Figure 2.18 Procédure 2: ébauche de l'heuristique 2

Dans ce qui suit, nous proposons une deuxième heuristique de sélection de bascules basée sur leurs contributions à la couverture de pannes globale du circuit. Les bascules sont d'abord ordonnées dans un ordre croissant selon leurs profondeurs logiques séquentielles. Ensuite, elles sont temporairement sélectionnées pour le *reset* partiel, une à la fois,

si leurs contrôlabilités à 1(0) sont inférieures à un seuil fixé à ".01". sCOP est alors invoqué pour calculer l'impact de chaque modification sur les contrôlabilités. La couverture de pannes est estimée à partir d'un modèle théorique déjà proposé par Shedletsky et McCluskey [122]. Lorsqu'une amélioration de la couverture de pannes est supérieure à un seuil "q", la bascule en question est retenue pour le *reset* partiel. Une autre bascule est ensuite sélectionnée et la procédure est répétée (voir Procédure 2). Dans la présente version, l'heuristique est mise en oeuvre avec "q = 0", c'est-à-dire que n'importe quelle amélioration de la couverture de panne implique une acceptation de la bascule pour le *reset* partiel.

2.6.3 Résultats expérimentaux

Pour les besoins de l'expérience, nous avons mis en oeuvre un outil de sélection des bascules à remettre zéro/un en langage "C". Dans une première phase, les bascules responsables des problèmes d'initialisation sont sélectionnées et remplacées par leur équivalents avec *reset*. Dans une deuxième phase, d'autres bascules responsables des problèmes de test sont sélectionnées et remplacées par leur équivalents avec *reset* en utilisant les heuristiques proposées ci-dessus.Pour valider nos heuristiques, nous avons réalisé plusieurs expériences sur un grand ensemble de circuits d'essai séquentiels [26]. Le simulateur de panne HOPE [74] fut utilisé avec des séquences de vecteurs de test pseudo-aléatoires de 2^{15} pour valider les résultats avant et après modification.

Le nombre total de bascules dans chaque circuit est reporté dans la colonne "DFF". Les couvertures de pannes rapportées par HOPE sur les circuits originaux à partir d'un état initial inconnu et d'un état initial connu "11.....1" (1 sur chaque bascule) sont reportées sous les colonnes "CPx" et "CP" respectivement. Il est intéressant de noter ici qu'il y a en général une grande différence entre "CPx" et "CP" dans le cas où le circuit a des problèmes d'initialisation. Les problèmes d'initialisation peuvent cependant se combiner à

des problèmes de test, pour causer des différences qui ne sont pas explicables uniquement par les problèmes d'initialisation. Par exemple, dans le cas du circuit s510, CPx est 0 alors que CP est 100%. Il semble que s510 n'a que des problèmes d'initialisation qui résultent en un CPx de 0. Par contre, dans le cas de s382, aucun problème d'initialisation n'a été détecté. s382 contient plusieurs problèmes de test et les couvertures de pannes obtenues dans les deux cas sont CPx = 13.28%, CP = 26.31% respectivement.

Tableau 2.4 Les résultats de la couverture de pannes tels qu'obtenus par Hope [74]

				Test 0		Test1		Test 2		PReSet	
Circuit	DFF	CPx	CP	CP0	R-FF	CP1	FF1	CP2	FF2	CP3	FF3
s27	3	100	100	100	0	100	0	100	0	-	-
s208	8	45.58	57.67	45.58	0	98.60	6	98.23	6	86.55	3
s298	14	82.79	86.03	82.79	0	87.82	2	92.58	1	97.13	2
s344	15	96.19	99.12	96.19	0	99.70	1	99.70	1	99.43	3
s349	15	95.71	98.57	95.71	0	99.14	1	99.14	1	100	5
s382	21	13.28	26.31	13.28	0	94.84	15	90.66	4	94.4	5
s386	6	69.01	69.01	69.01	0	96.15	3	96.15	3	90.93	1
s420	16	30.69	30.69	30.69	0	85.15	15	71.14	12	-	-
s444	21	12.65	22.78	12.65	0	91.63	14	83.40	4	95.00	2
s510	6	0	100	100	6	99.82	0	99.82	0	89.93	6
s526	21	9.36	19.82	9.36	0	83.19	14	65.00	4	88.93	3
s526n	21	9.40	19.89	9.40	0	82.44	14	65.24	4	-	-
s641	19	86.08	87.58	86.08	0	98.73	4	98.94	3	96.80	1
s713	19	81.58	82.61	81.58	0	92.53	4	92.65	2	90.22	1
s820	5	48.35	50.47	48.35	0	50.47	0	50.47	0	85.05	2
s832	5	47.47	49.65	47.47	0	49.65	0	49.65	0	-	-
s838	32	23.80	23.80	23.80	0	69.79	30	68.20	26	-	-
s953	29	8.34	99.07	99.07	19	81.89	2	81.89	2	91.49	1
s1196	18	98.39	98.39	98.39	0	98.39	1	98.39	0	98.72	3
s1423	74	56.37	58.87	56.37	0	67.41	22	63.89	11	90.07	2
s1488	6	74.96	75.57	95.19	6	95.19	0	95.19	0	97.38	1
s1494	6	74.03	74.63	96.31	6	96.31	0	96.31	0	98.08	1
s5378	179	62.35	66.02	66.06	22	88.99	94	86.59	30	85.3	17
s9234	228	0.45	26.38	28.84	174	28.84	30	28.84	30	-	-
s13207	669	6.41	21.20	26.78	478	29.43	36	24.83	15	-	-
s15850	597	0.73	19.23	21.82	286	31.76	233	26.05	36	53.20	25
s35932	1728	85.74	85.81	85.74	0	89.74	234	89.23	45	-	-
s38417	1636	3.61	15.12	13.28	973	43.59	70	28.90	56	-	-
s38584	1452	22.35	64.33	43.74	1107	45.44	29	45.58	4	-	-

•DFF: Nombre de bascules dans le circuit.

- •CPx and CP: couvertures de pannes du circuit original où l'état du départ est inconnu (CPx) et à partir de l'état (11...) (CP) respectivement.
- •Test 0: résultats obtenu en utilisant l'heuristique de la section 2.5.
- •Test 1: résultats obtenus avec l'heuristique 1.
- •Test 2: résultats obtenus avec l'heuristique 2.
- •PReSet: les meilleurs résultats obtenus par l'heuristique proposée dans [2] pour faciliter la génération algorithmique de CRIS

Dans le but de régler les problèmes d'initialisation, l'heuristique de la section 2.5 a été invoquée. Sous la colonne "Test0", nous avons reporté le nombre de bascules à remettre à zéro/un pour l'initialisation (colonne R-FF), ainsi que la couverture de pannes lorsque l'état initial est inconnu et que les lignes de *reset* sont utilisées comme des sources pseudo-aléatoires (Colonne CP0).

Les résultats montrent que l'utilisation des lignes de *reset,* pour alimenter les bascules retenues par l'heuristique d'insertion de points d'initialisation (section 2.5.1) durant le mode test avec des sources pseudo-aléatoires, peut résulter en une amélioration de couverture de pannes. Par exemple, dans le cas du circuit s1488, la couverture de pannes croît de 75.57% à 95.19%. Cependant, cette amélioration n'est pas systématique, en effet, des pertes de couvertures de pannes peuvent être observées dans le tableau. Par exemple, dans le cas de s38584, la couverture de pannes décroît de 64.33% à 43.74%. Ceci montre que le *reset* partiel peut avoir des effets négatifs sur la couverture de pannes dans certains cas. Par conséquent, nous pouvons imaginer une solution où nous désactivons le *reset* de quelques bascules utilisées pour l'initialisation durant le mode test. Ce problème n'est pas considéré ici, il peut être l'objet d'une recherche future.

Les résultats obtenus après modifications avec les heuristiques 1 et 2 sont reportés sous les colonnes "Test1" et "Test2" respectivement. Les nombres de bascules à remettre à

zéro/un obtenus par chaque heuristique sont reportés sous les colonnes "FF1" et "FF2". Les résultats des couvertures de pannes sont reportées sous les colonnes "CP1" et "CP2".

Les résultats montrent une augmentation systématique de la couverture de pannes. Par exemple, dans le cas de s208, la couverture de panne croît de 57.67% à 98.60% avec l'heuristique 1 et jusqu'à 98.23% avec l'heuristique 2. Le nombre de bascules remises à zéro/un dans ce cas est 6. Nous pouvons également constater que le nombre de bascules sélectionnées par l'heuristique 2 est souvent plus petit que celui des bascules sélectionnées par l'heuristique 1. Cependant, ceci est associé à une perte de couverture de pannes dans certains cas. Par exemple, dans le cas de s5378, le nombre additionnel de bascules remise à zéro/un avec l'heuristique 1 est 94 et la couverture de pannes est 88.99%. Avec l'heuristique 2, ce nombre est 30 avec une couverture de pannes de 86.594%.

Nous avons également reporté au tableau 2.4, sous la colonne PReSet, les meilleurs résultats publiés dans la littérature. PReSet [2] est une technique de *reset* partiel destinée au test algorithmique. Un petit ensemble de bascules à remettre à zéro/un sont sélectionnées pour améliorer la génération algorithmique des vecteurs de test. Les couvertures de pannes dans ce cas sont obtenues par le générateur (ATPG) CRIS [110]. Une comparaison entre nos résultats et ceux obtenus par PReSet montre que nos heuristiques produisent de meilleurs couvertures de pannes dans plusieurs cas. Par exemple, dans le cas du circuit s208, heuristique 2 produit une couverture de pannes de 98.24% avec 6 bascules remises à zero/un. De l'autre côté, avec PReSet, on obtient 86.55% de couverture de pannes avec 3 bascules remises à zero/un. Cependant, ceci n'est pas systématique. Par exemple, dans le cas de s526, PReSet produit 88.93% de couverture de pannes avec 3 bascules remises à zero/un alors que l'heuristique 1 donne 83.19% avec 14 bascules remises à zero/un. En conclusion, en moyenne, le *reset* partiel proposé ici est aussi performant que PReSet.

Cependant à notre connaissance, aucune autre heuristique de *reset* partiel n'a été proposée dans la littérature dans un contexte de test pseudo-aléatoire.

CHAPITRE 3

La probabilité de transition: une mesure de testabilité efficace

3.1 Introduction

Les mesures de testabilité approximatives constituent un bon compromis entre la faculté de prédire et de localiser les problèmes de test d'un côté et leurs complexités de calcul d'un autre côté. Dans un contexte de test pseudo-aléatoire, les mesures COP par exemple furent efficacement utilisées par plusieurs auteurs dans le cas des circuits combinatoires [119][145]. Malheureusement, dans le cas des circuits séquentiels, les mesures de testabilité classiques (sCOP) présentent d'autres limitations reliées à la nature séquentielle de ces circuits. Nous insistons ici sur le fait qu'il ne s'agit pas de limitations reliées aux hypothèses d'indépendance souvent utilisées dans les mesures de testabilité classiques. Ces limitations furent largement étudiées dans la littérature [10][48][61][88][117]. Nous allons plutôt nous concentrer dans ce chapitre sur d'autres types de limitations reliées à la nature séquentielle des circuits analysés. Nous proposerons ainsi une nouvelle mesure de testabilité nommée mobilité. Cette mesure, dont la complexité de calcul demeure linéaire par rapport au nombre de portes d'un circuit, est capable de détecter des problèmes de testabilité très importants dans le cas des circuits séquentiels.

La prédiction des probabilités de transistion a fait l'objet de plusieurs publications dans le domaine de l'estimation de la dissipation de puissance pour les circuits CMOS

[51][80][86][89][97][137][138]. En effet, dans un circuit CMOS convenablement conçu, la dissipation de puissance est essentiellement due à l'activité des noeuds internes. Dans un contexte de test pseudo-aléatoire, où l'activité des noeuds internes est anormalement élevée, l'estimation de la puissance dissipée prend une importance particulière. En effet, un échauffement excessif durant le mode test risque de brûler un composant sous test, alors que nous désirons simplement le tester.

Ce chapitre est composé de deux articles. Dans le premier, soumi pour publication au "IEE Transactions", nous introduisons le concept de mesure de mobilité et nous étudierons ses capacités à prédire et à localiser les problèmes de test dans les circuits séquentiels. Dans le deuxième article, soumi pour publication au "IEEE Transactions on VLSI" nous montrerons comment utiliser le concept de mobilité pour estimer la consommation de puissance dans les circuits séquentiels.

3.2 Résumé de l'article de la section 3.3

La mobilité est basée sur le concept de probabilité de transition. Pour chaque noeud, quatre mobilités qui représentent les probabilités de toutes les transitions possibles sont définies ($m_{01}^{l}(t)$, $m_{10}^{l}(t)$, $m_{00}^{l}(t)$, $m_{11}^{l}(t)$). Les mobilités sont reliées aux nombres sCOP comme suit:

$$c_0^l = m_{00}^l + m_{10}^l$$
$$c_1^l = m_{01}^l + m_{11}^l$$

Notons ici que les mobilités sont en réalité similaires aux contrôlabilités. Nous avons toujours besoin des observabilités pour en faire un système de mesures de testabilité complet. Pour chaque type de porte logique, nous pouvons calculer les mobilités de la sortie en fonction de celles des entrées, par le biais de règles de calcul simples similaires à celles qui sont utilisées pour le calcul des nombres COP. Par exemple, dans le cas d'une porte

AND à deux entrées, les mobilités de la sortie "S" sont reliées aux mobilités des entrées {A, B} comme suit:

$$m_{11}(S) = m_{11}(A) \times m_{11}(B)$$
$$m_{01}(S) = m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{11}(B) + m_{11}(A) \times m_{01}(B)$$
$$m_{10}(S) = m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B)$$
$$m_{00}(S) = 1 - (m_{11}(S) + m_{10}(S) + m_{01}(S))$$

En réalité, seulement deux mobilités parmi les quatres doivent être calculées, les deux autres peuvent être déduites à partir du système de deux équations linéaires ci-dessous:

$$m^{l}_{01}(t) = m^{l}_{10}(t)$$
$$m^{l}_{01}(t) + m^{l}_{10}(t) + m^{l}_{00}(t) + m^{l}_{11}(t) = 1$$

Comme nous le verrons, dans certaines situations, des contrôlabilités acceptables peuvent être associées avec des mobilités faibles, ce qui peut engendrer un problème de testabilité. Dans le cas d'un compteur binaire par exemple, les probabilités limites (contrôlabilités) sont de 0,5 pour tous les bits. Cependant, les séquences requises pour tester ces bits sont de longueurs différentes. Par exemple, dans un compteur à 5 bits dont l'entrée "enable" est excitée par un signal pseudo-aléatoire, le bit le moins significatif transite d'un état à un autre à chaque 2 cycles en moyenne. Cependant, le temps moyen entre 2 transitions consécutives de même polarité sur le bit le plus significatif est de 2^6 cycles.

Pour des fins de comparaison entre les contrôlabilités et les mobilités, nous avons conduit un ensemble d'expériences sur plusieurs circuits d'essai séquentiels. Les résultats nous ont montré que les mobilités sont systématiquement meilleures que les contrôlabilités relativement à la capacité de prédire et de localiser les sites de problèmes de test.

3.3 Mobility Measures for Pseudo-Random Testing of Sequential Circuits

M. Soufi, Y. Savaria, B. Kaminska, F. Darlay

ABSTRACT

In this paper, probabilistic testability measures for sequential circuits are analyzed. Shortcomings associated with the sequential nature of digital circuits are highlighted. These shortcomings are different from those related to the logical independence assumption between internal nodes, often used in approximate testability measures (ATM). In order to overcome these shortcomings, a new testability measure, called mobility, is introduced. Mobility is based on the concept of transition probability calculations. This concept has been widely used in the past for estimating power consumption in integrated circuits. In this paper, we will show that the mobility measure is of great interest in assessing the testability of sequential circuits. Indeed, it models their testability with pseudo-random vectors more accurately. Experimental results demonstrating the superiority of the mobility in predicting controllability detection problems are also presented. The fraction of hard-to-detect faults which escape detection in testability analysis using the mobilities is systematically lower than that which escape detection using controllabilities.

3.3.1 Introduction

The increased complexity of VLSI circuits leads to a gradual reduction of accessibility to internal nodes. As a consequence, the testing of digital systems is a recurring challenge, despite significant advances in the field. This challenge is likely to remain, as long as the complexity of digital systems keeps increasing. For instance, Brewer et al. recently reported the introduction of a 26-million-transistor microprocessor [23], which clearly confirms that the growth of complexity will remain for many years to come. Moreover, it

is well known that the quality of a chip is directly dependent on the percentage of defective chips which escape detection. Several studies have confirmed that the reject ratio[1] depends almost linearly on the fault escapes, defined as (1 - fault coverage) [11]. In this context, it is more important than ever to predict testing difficulties early in the design process. This was the purpose of the numerous algorithms proposed in the past for estimating testability [54][64][144]. Many authors have also explored pseudo-random testing as a solution to the testing complexity problem. Most pseudo-random testing methods were essentially devoted to combinational circuits. Recently, several authors [18][75][129] have proposed probabilistic testability measures for sequential circuits. However, as we will see in this paper, these measures are blind to some important testing problems in sequential circuits.

Historically, pseudo-random test methods have been essentially developed for combinational circuits. This is largely because the specific orderings of the circuit operations required to test some sequential faults are prohibitively difficult to produce using a pseudo-random source. Our paper formulates concrete proposals to help solve this problem. In a recent past, several papers related to the subject of testing sequential circuits with pseudo-random vectors have been published in the literature [32][75][91][118]. In [118], for instance, a test point insertion technique for sequential circuits is proposed and testability analysis is used to detect areas of poor testability. This technique is well suited to implement built-in self-test with pseudo-random vectors. In [32], another pseudo-random technique which can be used with sequential circuits is proposed. In that paper, a symbolic computation technique is used to compute testability for circuits under the par-

1. Reject ratio is defined as $$\text{Reject ratio} = \frac{\text{\# of defective chips tested as good}}{\text{\# chips tested as good}}$$

tial-scan based BIST. Moreover, the first commercial BIST tool for partial scan designs, announced by AT&T, is already on the market place [75].

In the present paper, limitations of classical testability measures will be highlighted. These limitations are not associated with the independence assumption usually used in the computation of these measures. This paper focuses on the shortcomings related to the sequential nature of digital circuits. For instance, we will show that good controllabilities may be associated with poor activity (transitions from 0 to1 and vice versa) in sequential circuits. In this case, prediction of test length using the controllabilities may escape some testing problems, resulting in less effective tests. A new testability measure, called mobility, is proposed to partly overcome these limitations. It will be shown that the number of pseudo-random resistant faults which escape detection with these measures is significantly smaller with mobilities than with controllabilities.

The rest of the paper is organized as follows. In section 3.3.2, some limitations of classical testability measures are highlighted. In section 3.3.3, mobilities are introduced for combinational and sequential circuits. Section 3.3.4 reports results of experiments conducted on a large subset of the ISCAS 89 sequential benchmarks to evaluate the efficiency of the mobility measures. Our main conclusions and remarks are summarized in section 3.3.5.

3.3.2 Limitations of Classical Testability Measures

In pseudo-random testing, largely used in built-In self-test (BIST) applications, detection probabilities, defined as the probability of detecting a given fault using randomly generated test vectors, are usually estimated by means of signal probabilities. These measures, based on signal probabilities, originally defined for combinational circuits

[27][28], were recently generalized to sequential circuits by several authors [18][75][129]. In [18], the initiability measure was introduced to cover the initialization phase of sequential circuits. Lin et al. [75] proposed a new testability measure calculation method, which decomposes a circuit into smaller blocks and then computes their internal signal probabilities, one block at a time. However, several approximations were considered in [75], approximations which may result in inaccuracies. In [129], the authors proposed an iterative testability measure calculation, based on the assumption that sequential circuits can be modeled using ergodic Markov chain processes.

All the testability measures mentioned above estimate the probability of setting internal nodes to 1 or 0 by applying random vectors to primary inputs. However, even if exact calculation methods are used to estimate the measures, they will still suffer from some inaccuracies related to the time dependency. For instance, Table 3.1 shows examples of signals with equal steady-state probabilities (c_0 and c_1). However, these signals exhibit very significant differences from a testability standpoint. They have different steady-state transition probabilities. In the rest of this section, after a brief introduction to classical testability measures in sequential circuits as proposed in [129], which is necessary for understanding the rest of the paper, we will show that these measures have important practical shortcomings. The purpose of the mobility measures proposed in this paper is to address these shortcomings.

Table 3.1 Classical controllabilities do not uniquely describe a signal in sequential circuits

Example	c_0	c_1	trans 0->0	trans 0->1	trans 1->0	trans 1->1
	1/2	1/2	0	1/2	1/2	0

Table 3.1 Classical controllabilities do not uniquely describe a signal in sequential circuits

Example	c_0	c_1	trans 0->0	trans 0->1	trans 1->0	trans 1->1
	1/2	1/2	1/4	1/4	1/4	1/4
	1/2	1/2	1/3	1/6	1/6	1/3
	1/2	1/2	3/8	1/8	1/8	3/8

3.3.2.1 Classical Testability Measures for Sequential Circuits

Figure 3.1 shows an example of a sequential circuit with its corresponding ergodic Markov chain process. The numbers in brackets are the primary input values which cause a specific transition, and the corresponding fraction represents its occurrence probability. Let $\pi_i(t)$ be the probability that the Markov chain is in state "i" after "t" cycles (i.e. after the application of "t" pseudo-random input patterns to the circuit). A state probability vector of a sequential circuit with "n" flip-flops (2^n states) is defined as $\Pi(t) = \left(\pi_0, \pi_1, \ldots\ldots \pi_{(2^n-1)}\right)$. Then, $\Pi(t)$ can be obtained using the initial probability vector $\Pi(0)$ and the transition probability matrix "P" as follows:

$$\Pi(t) = \Pi(0) \times P^t$$

where P is defined as the matrix of individual transition probabilities in one cycle between states "i" and "j":

$$(P = [p_{ij}]),\ i,j = 0, 1, 2 \ldots\ldots (2^n - 1).$$

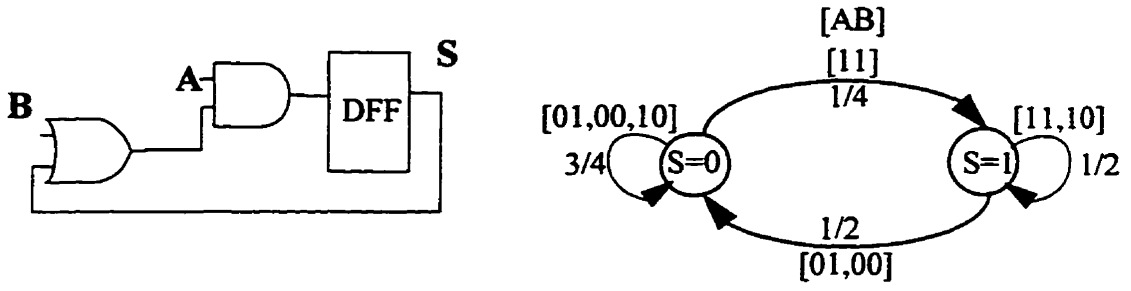


Figure 3.1 Example of a sequential circuit with its corresponding Markov chain process

The ergodicity assumption guarantees the existence of the limiting probability vector defined as

$$\Pi = \lim_{t \to \infty} \Pi(t)$$

For instance, in the example of Figure 3.1, the limiting probabilities are

$$c_0 = \frac{2}{3}$$

$$c_1 = \frac{1}{3}$$

Note that in this example, the limiting probabilities are controllabilities of "S" to 0 and 1. For circuits with more than one flip-flop, the controllabilities to 0 (1) of a flip-flop output may be computed by adding all the limiting probabilities of the states where this flip-flop is at 0 (1) [129].

3.3.2.2 Shortcomings of Sequential Testability Measures

Consider the example in Figure 3.2. The computation of c_0 and c_1 leads to the same numerical result as in Figure 3.1 ($c_0 = \frac{2}{3}$, $c_1 = \frac{1}{3}$).

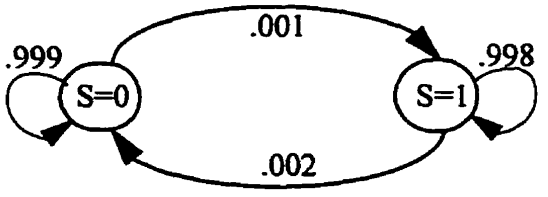


Figure 3.2 State transition graph for which the limiting probabilities are a misleading indicator of testability

However, assuming pseudo-random testing, it is obvious that the circuit corresponding to the FSM diagram in Figure 3.2 is more difficult to test than the first one. Indeed, the transitions from one state to another are unlikely to occur as rapidly as in the first example, and the length of the pseudo-random test covering all states would be much greater. There is a great deal less activity (i.e. transitions between different states). Note that activity is a desirable characteristic in pseudo-random testing. Thus, limiting probabilities may be completely misleading if used as a direct measure of testing difficulty.

In order to estimate this activity, let us calculate the average time the Markov chain remains in either of the two states. Let U and D denote the average time the Markov chain remains in states 1 and 0 respectively.

Theorem[1] 1: In a two-state machine, the average times U and D that a Markov chain remains in state 0 and 1 are respectively given by

$$U = \frac{1}{P_{10}} \qquad D = \frac{1}{P_{01}}$$

A proof of this theorem is provided in Appendix 4.

For instance, in the example of Figure 3.1, we find

1. This theorem is a direct application of Markov chain theory [106]

$$U = 2, \qquad D = 4$$

and the rate at which each transition occurs is

$$\frac{1}{U+D} = \frac{1}{2+4} = \frac{1}{6}$$

Thus, in the example of Figure 3.2, U, D and the rate at which each transition occurs are respectively

$$U = 500, \qquad D = 1000, \qquad \frac{1}{U+D} = \frac{1}{1500}$$

This means that, on average, the second Markov chain (Figure 3.2) stays 1000 time units in state 0 before making a transition to state 1, and it stays there 500 time units before returning to state 0, which is very likely to cause a testing problem in a system targeted for pseudo-random testing. For instance, some logic connected to the output of that state machine would be exercised at a very low frequency. Note that this is so in spite of good limiting probabilities (c_0=2/3 and c_1=1/3).

This problem arises in a very popular digital block, the binary counter. Indeed, even if the limiting probabilities are as good as .5 for all bits, it is clear that the sequences needed to test these bits are of different lengths. For instance, in a 5-bit counter excited with an equiprobable pseudo-random signal on its enable input, a transition occurs in the least significant bit every two cycles on average, however, in the most significant bit, the average time between two successive transitions of the same polarity is 2^6 cycles. Table 3.2

summarizes the classical controllabilities and the average transition rates for each bit of the 5-bit counter.

Table 3.2 Controllabilities and transition rates for a 5-bit counter with a pseudo-random enable

FF	c_0	c_1	0->0	0 -> 1	1 -> 0	1 -> 1
enable	1/2	1/2	1/4	1/4	1/4	1/4
bit 1	1/2	1/2	1/4	1/4	1/4	1/4
bit 2	1/2	1/2	3/8	1/8	1/8	3/8
bit 3	1/2	1/2	7/16	1/16	1/16	7/16
bit 4	1/2	1/2	15/32	1/32	1/32	15/32
bit 5	1/2	1/2	31/64	1/64	1/64	31/64

3.3.3 Mobility Measures

In order to overcome the above limitations of existing sequential testability measures, we propose here to use a set of measures called mobilities. Mobilities are analogous to the classical controllabilities. Mobilities must be combined with observability to make a complete set of testability measures for estimating detectability. These measures are based on transition probability calculations. They are defined as the probability that a transition occurs on a line "*l*" in a given circuit. This transition can be up (0 -> 1) or down (1 -> 0). Moreover, in order to get a complete set of transitions, we define transitions between identical states (0 -> 0 and 1 -> 1). We denote these mobilities $m_{01}^{l}(t)$, $m_{10}^{l}(t)$, $m_{00}^{l}(t)$, $m_{11}^{l}(t)$, where, for instance, $m_{01}^{l}(t)$ is defined as

$$m_{01}^{l}(t) = \text{probability that } l(t) = 0 \text{ and } l(t+1) = 1$$

The other mobilities are defined similarly. In the following, we first introduce the mobility measures in combinational circuits, then we will generalize them to sequential circuits. In section 3.3.3.3, we will show how to propagate the measures in a digital circuit.

3.3.3.1 Combinational circuits

In combinational circuits, the two events

$$l(t) = 0$$

and

$$l(t+1) = 1$$

are mutually independent, and of course they are time-independent if their corresponding input vectors are independent. Thus, we can obtain

$$m_{01}^{l}(t) = m_{01}^{l} = c_0(l) \times c_1(l) \tag{3.1}$$

In combinational circuits, the mobilities can be directly computed from the classical controllabilities. As a consequence, the classical controllabilities are sufficient to characterize the testability of combinational circuits. Note here that approximate calculation methods [61], based on simplified assumptions, such as the independence between internal nodes in the circuit, may affect the quality of these measures. However, if no such assumption is made, the measures are accurate for combinational circuits.

3.3.3.2 Sequential Circuits

In general sequential circuits, the two events mentioned above

$$l(t) = 0$$

and

$$l(t+1) = 1$$

are neither mutually independent nor time-independent. As a consequence, equation (3.1) is no longer valid.

A sequential circuit containing "n" flip-flops may be modeled with a 2^n state space Markov chain process. In this case, each transition (0 -> 1) on a flip-flop output corresponds to a transition from the half state space "sub_0", where this flip-flop output is at "0", to the other half "sub_1", where the flip-flop output is at "1".

Lemma 1: Mobility measures in ergodic sequential circuits always exist, and their values are given by

$$m_{xy}(l) = \sum_{(j \in sub_y)} \sum_{(i \in sub_x)} \pi_i P_{ij} \tag{3.2}$$

with $0 \leq m_{xy} \leq 1$ and $(x,y) \in \{0,1\}^2$.

The proof of this Lemma is a direct consequence of ergodic Markov chain processes and the mobilities definition. A sketch of the proof is provided in Appendice 5.

One should note here that this lemma has a theoretical value, since it states the existence (and uniqueness) of the mobilities. However, computing the mobilities from Markov chain processes (equation 3.2) is a time-consuming process for large circuits. Indeed, handling the double summation of equation 3.2 has a complexity of N^2, where $N = 2^n$ is the total number of states, assuming there are "n" state bits. Therefore, the complexity is proportional to 2^{2n}. In the next section, we will propose a simple gate-level algorithm for computing approximations of the mobilities. This algorithm has a complexity that is linear with respect to the number of gates times the number of computation iterations.

Before going further, let us mention that classical controllabilities can be computed from mobilities, but not vice versa. As a consequence, it is sufficient to specify the mobilities. The relations between classical controllabilities and mobilities are as follows:

$$\begin{aligned} c_0^l &= m_{00}^l + m_{10}^l \\ c_1^l &= m_{01}^l + m_{11}^l \end{aligned} \tag{3.3}$$

NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was microfilmed as received.

This reproduction is the best copy available.

UMI

identified, and two escape probabilities, one for each case (controllability and mobility) were computed. The escape probabilities for the controllabilities and the mobilities are respectively defined below.

$$ESC_{control} = \frac{CF}{T} \tag{3.6}$$

$$ESC_{mobility} = \frac{MF}{T} \tag{3.7}$$

where

1.CF: # of faults undetected with fault simulation declared detected according to their detectabilities, computed based on their controllabilities and observabilities.

2.MF: # of faults undetected with fault simulation declared detected according to their detectabilities, computed based on their mobilities and observabilities.

3.T: Total number of undetected faults according to fault simulation.

Table 3.5 Escape probability in the case of mobility and controllability measures

Circuit name	#Redundant faults(10^7)	#Undetected faults(2^{15})	DT = .1					DT = .01					DT = .001				
			$ESC_{mobility}$		$ESC_{control}$		M/C %	$ESC_{mobility}$		$ESC_{control}$		M/C %	$ESC_{mobility}$		$ESC_{control}$		M/C %
			#	%	#	%		#	%	#	%		#	%	#	%	
s27	0	0	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s208	74	91	0	0	2	12	0	3	18	4	24	33	3	18	4	24	33
s298	35	43	1	12	5	62	19	3	38	6	75	51	5	62	6	75	83
s344	3	3	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s349	5	5	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s382	279	294	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s386	80	119	3	8	9	23	35	8	21	15	38	55	10	26	17	44	59
s420	226	250	0	0	5	21	0	3	12	6	25	48	4	17	6	25	68
s444	341	366	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s510	0	0	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s526	416	445	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s526n	414	443	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s641	58	58	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s713	101	101	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s820	388	421	4	12	13	39	31	20	61	26	79	77	26	79	27	82	96
s832	406	438	4	12	15	47	26	22	69	26	81	85	27	84	27	84	100
s838	545	561	0	0	4	25	0	3	19	5	31	61	4	25	5	31	81
s953	10	10	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s1196	3	20	3	18	3	18	100	6	35	6	35	100	6	35	7	41	85
s1423	309	623	4	1	32	10	10	21	7	65	21	33	38	12	84	27	44
s1488	138	363	64	28	124	55	51	139	62	146	65	95	144	64	151	67	96
s1494	154	382	67	29	129	57	16	142	62	150	66	94	147	64	155	68	94
s5378	1411	1564	3	2	14	9	22	4	3	14	9	33	8	5	56	37	14
s9234	2896	2896	0	0	0	0	-	0	0	0	0	-	0	0	0	0	-
s13207*	-	7444	508	7	902	12	58	952	13	1426	19	68	1413	19	2051	28	68
s15850*	-	9294	505	5	871	9	56	824	9	1467	16	56	1702	18	2264	24	75

Table 3.5 Escape probability in the case of mobility and controllability measures

Circuit name	#Redundant faults(10^3)	#Undetected faults(2^{15})	DT = .1					DT = .01					DT = .001				
			$ESC_{mobility}$		$ESC_{control}$		M/C %	$ESC_{mobility}$		$ESC_{control}$		M/C %	$ESC_{mobility}$		$ESC_{control}$		M/C %
			#	%	#	%		#	%	#	%		#	%	#	%	
s5841*	-	25853	712	3	1033	4	75	2273	9	3194	12	75	4221	16	5869	23	7
s38584*	-	12946	838	6	1656	13	46	2428	19	4546	35	54	5183	40	8009	62	65

1.#: number of faults that escape detection with the mobilities (controllabilities)

2.%: $ESC_{mobility}$ ($ESC_{control}$)

3.M/C: $ESC_{mobility} / ESC_{control}$

4.# Undetected faults: the number of faults undetected according to HOPE after applying 2^{15} test patterns. These numbers include the redundant faults for circuits marked with *.

In Table 3.5, the escapes $ESC_{mobility}$ and $ESC_{control}$ are reported for three different detectability thresholds (DT=.1, DT=.01 and DT=.001). Selecting an appropriate detection threshold is a difficult task. Indeed, a detection threshold which gives optimal results with some circuits may fail for other circuits. There are several application dependent contradicting requirements that should be taken into account when determining a detection threshold. The maximum length of the pseudo-random sequence used, the minimum fault coverage required, and, in the case of test insertion, the maximum overhead tolerated. As in many heuristic methods, the user should experiment with the optimal value of the parameter. A tool could also be programmed to try several values of the detection threshold. The existence of intrinsic limitations to the accuracy of heuristic testability measures has been recognized a long time ago. In [117] for instance, the author showed that good detectability does not guarantee good testability. Therefore, all methods using detectability as a guidance heuristic may fail for some circuits. Nevertheless, detectability is widely used in spite of the fact that its accuracy is not guaranteed.

As expected, Table 3.5 shows that $ESC_{mobility}$ is systematically lower than $ESC_{control}$. In the case of s208 with DT = .1, for instance, 12% of the faults that are found undetected by Hope were not declared undetected by the controllability measures. Howe-

ver, this percentage is 0% when the mobility measures are used. As a consequence, in s208, the mobility measure captured all irredundant pseudo-random resistant faults (undetected faults). Unfortunately, this is not the case for all benchmark circuits. Some resistant faults may escape to the mobility measures, as in s1494, where 29% were not captured. Moreover, as shown in Table 3.5, in general, the ratio M/C between $ESC_{mobility}$ and $ESC_{control}$ decreases as the detection threshold (DT) increases. For instance, in the case of s820, this ratio decreases from 96% to 77%, and then to 31% as its corresponding detection threshold increases from .001 to .01 to .1 respectively. Thus, the mobilities seems to be more efficient than the controllabilities as the detection threshold is increased.

The previous results show that the mobility measures are systematically better than the classical controllability measures. The difference is significant, because it is well known that the closer we are to perfect fault coverage, the more difficult it is to increase it. However, even if these measures were capable of detecting all testability problem sites in several circuits, they seem to be less efficient for other circuits. For instance, in the case of s1488 with DT = .01, 62% escape has been observed. This is related to the fact that other forms of testability problems may exist. For instance, the pathologies mentioned in [49] is a testability problem that cannot be detected with the proposed measures.

3.3.5 Conclusions

In this paper, testability measures for sequential circuits were studied. We demonstrated that existing testability measures have shortcomings which are not associated with the node independence assumption that is usually considered in their computation. This node independence issue has already been addressed in many papers. This paper thus focused on the shortcomings associated with the sequential nature of digital circuits. Indeed, these problems do not exist in purely combinational circuits. We showed that good controllabili-

ties may be obtained even if the activity (transition from 0 to 1 or vice versa) is very low. In this case, predicted detection probabilities using controllabilities may escape many testing problems, and thus less effective tests may result. We therefore proposed the mobility measure to quantify the activity of all internal nodes. Experimental results demonstrating the superiority of the mobility in predicting hard-to-detect faults were also presented. Indeed, the fractions of undetectable faults which escape detection using the mobilities were systematically lower than those which escape detection using controllabilities. These significant improvements over the literature could be very helpful in the difficult and time consuming task of approaching ideal fault coverage.

It should also be noted that power dissipation in CMOS digital circuits is directly proportional to the average activity of their nodes. Therefore, a good mobility, which is a desired characteristic in pseudo-random testing, as mentioned above, may cause power dissipation problems. This problem of managing power dissipation while testing is another clear potential application of the mobility measures. An analysis of this issue shall be presented elsewhere.

3.4 Résumé de l'article de la section 3.5

Les problèmes de fiabilité reliés à la dissipation excessive de puissance et à l'électromigration deviennent de plus en plus une préoccupation importante pour les concepteurs des circuits VLSI modernes. Historiquement, les concepteurs étaient principalement préoccupés par la surface et les performances de leurs circuits, ainsi que par la fiabilité et son impact sur les coûts. Cependant, avec l'augmentation accrue des complexités des circuits VLSI modernes et de leurs fréquences d'utilisation, le problème de la dissipation de puissance s'est imposée comme une autre contrainte de conception aussi importante que les contraintes classiques (surface, performance). De nos jours, un concepteur n'a générale-

ment plus le luxe de remettre la gestion de la dissipation de puissance comme une tâche post conception. Par conséquent, l'obtention d'informations sur la consommation de puissance très tôt durant le cycle de conception est devenue nécessaire pour minimiser et gérer de façon adéquate la consommation de puissance.

Comparée à d'autres technologies, la technologie CMOS fut historiquement considérée comme une technologie à basse consommation de puissance. Cependant, avec les complexités et les fréquences d'utilisation des circuits VLSI modernes, même avec la technologie CMOS, la consommation de puissance est de plus en plus problématique. Récemment, des technologies CMOS à basse tension ont été proposées pour réduire la consommation de puissance. En effet, pour les circuits CMOS, la puissance dissipée est reliée à la tension par une relation quadratique. Par conséquent, une diminution de la tension par un facteur "n" réduit la consommation de puissance par un facteur "n^2".

Dans la technologie CMOS, il existe principalement trois sources de dissipation de puissance:

1. la dissipation de charge et de décharge: c'est la dissipation qui est due aux charges et décharges de la capacité des noeuds. Généralement, la dissipation de charge et décharge est la plus importante composante de la dissipation de puissance dans les circuits CMOS.
2. la puissance de court-circuit: c'est la puissance dissipée dans chaque porte quand les réseaux P et N conduisent simultanément.
3. la puissance de fuite: c'est la puissance dissipée à cause des fuites sous le seuil des transistors et des courants de fuite dans les jonctions P-N polarisées en inverse. La puissance de fuite représente la plus grande partie de la puissance dissipée au repos.

Durant le fonctionnement normal d'un circuit, les deux premières sources contribuent à plus de 99% de la puissance dissipée totale. Or ces deux composantes sont directement reliées aux transitions des noeuds internes. Par conséquent, la consommation de puissance dans les circuits CMOS est fortement reliée à l'activité des noeuds internes.

Dans l'article qui suit, nous utiliserons les mesures de mobilité introduites dans l'article précédent pour estimer la consommation de puissance dans les circuits CMOS séquentiels ainsi que combinatoires. L'originalité de ce travail se situe à notre avis dans l'utilisation d'une procédure de calcul dont la complexité en fonction du nombre de portes des circuits est linéaire. Ainsi, nous montrerons que les hypothèses simplificatrices pour assurer cette linéarité n'introduisent pas d'erreurs importantes. Ceci est particulièrement vrai et significatif pour des circuits de grande complexité. En effet, on évite une croissance excessive des temps de calcul et l'erreur relative sur l'estimation de la puissance dissipée moyenne décroît comme la racine carrée du nombre total de portes dans un circuit.

3.5 An Iterative Calculation Method for Efficient Estimation of the Activity in Large Sequential Circuits

M. Soufi, Y. Savaria and B. Kaminska

This paper addresses the problem of calculating transition probabilities, which are widely used for estimating the average power consumption in CMOS logic circuits. An efficient framework for computing these probabilities for large sequential circuits is proposed. This framework can handle several estimation methods proposed in the literature. The proposed framework was implemented with an approximate method for estimating transition probabilities in large sequential circuits. Our method is shown to be several orders of magnitude faster than previously reported procedures. Indeed, its complexity is linear with respect to the total number of gates. Therefore, it becomes possible to estimate the average power consumption of large sequential machines such as the ISCAS s38584 in a reasonable time. Moreover, we will show that the inaccuracies introduced by the adopted approximations are not very large, and for large circuits, they tend to average out.

3.5.1 Introduction

Due to the growing complexity of modern digital circuits, and the continuing increase in their clocking rates, reliability problems related to excessive power consumption and electromigration are becoming a major concern. Informations on power requirements are needed in the design process for minimizing and better managing power consumption. Historically, VLSI designers were primarily concerned with performance, area, reliability and their effects on costs. However, power consumption is gradually joining area and timing optimizations as a parameter of primary significance in circuit design. Recently, some CAD tools for estimating and optimizing power consumption have been introduced on the market [130]. These tools offer probabilistic as well as stimuli-based power estimations. However, little information on their implementations are available in the literature, which makes comparison with the method proposed in this paper difficult.

Compared to other technologies, CMOS logic circuits have historically been considered low power devices. However, the trend towards ever increasing clock frequencies can null this inherent advantage of CMOS circuits. Recently, low voltage CMOS technologies have been proposed to alleviate the power consumption problems. The rationale behind this strategy is related to the square relationship between the power consumption and the supply voltage. As a consequence, reducing V_{dd} results in interesting power consumption reductions. Nevertheless, with the continuing increases of speeds and circuit complexities, the pressure on managing the power dissipation budget is becoming more and more important. Before going further, let us review the most important components of power dissipation in CMOS circuits.

3.5.1.1 Power Dissipation in CMOS Circuits

There are 3 important components contributing to power dissipation in CMOS circuits as shown in Figure 3.5.

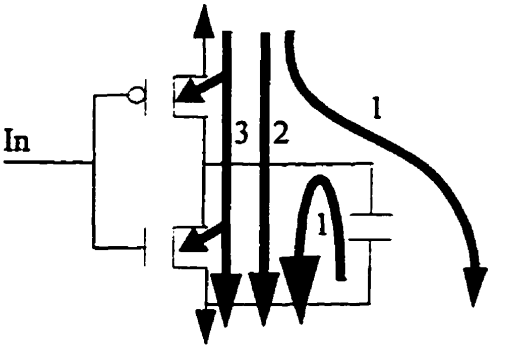


Figure 3.5 Components of power dissipation in CMOS circuits

1. Load capacitance charging-discharging power: This is the power dissipated during the charge and discharge of the load capacitance associated with each gate. Usually this is the main source of power dissipation.
2. Short circuit power: This is the power dissipated within each gate when the n and the p transistor networks simultaneously conduct.
3. Leakage power: This is the power dissipated due to sub-threshold leakages and due to the current flow through the reverse-biased P-N junctions. It accounts for the majority of the power dissipation when the circuit is inactive.

During normal operation, the first two components generally account for more than 99% of the power consumption in current CMOS technologies. Therefore, power consumption is highly dependent on internal node activity, which is determined by the applied input vector set.

3.5.1.2 Estimating the Switching Activity

Estimating the average power consumption in normal operation requires determining when and how often transitions occur at all nodes. The switching activity may be estima-

ted with logic simulations using functional vectors, however, this is an expensive process, because a large number of input patterns may be needed to accurately compute the power consumption [95][96]. Probabilistic methods have been proposed and studied by several researchers [51][80][86][89][97][137][138] as an alternative for efficiently estimating power consumption in digital circuits. These methods do not need any input vectors, and they avoid time-consuming simulations. Recently, a hybrid method combining a simulation-based technique and a probabilistic based technique was proposed in [31].

In order to be exact, spatial as well as temporal correlations between the nodes of a circuit should be taken into account. The spatial correlation accounts for the degree of dependence between nodes. For instance, in Figure 3.6, the nodes F and G are "spatially correlated". They both depend on B. Moreover, the node "S" at time "t+1" is temporally correlated to the nodes "G" and "F" at time "t". Another kind of time dependence not due to the sequential nature of the circuits is the difference between the delays through different combinational paths, which may result in additional switching activities. For instance, if the delay from B to E to F is different than the delay from B to G, spurious transitions can be generated. These transitions are usually filtered out by the latches and flip-flops in normal operation of a circuit, but they do contribute to the exact value of the power consumption. There is another situation where the timing informations are required. Indeed, in sequential circuits, multicycle paths are sometimes encountered. Such paths are allowed to stabilize in more than one clock period with the inputs kept constant during that interval. Considering multicycle paths as single cycle paths may result in inaccurate estimates.

In this case, the timing informations (the number of cycles allowed to stabilize each path) are required to produce accurate results.

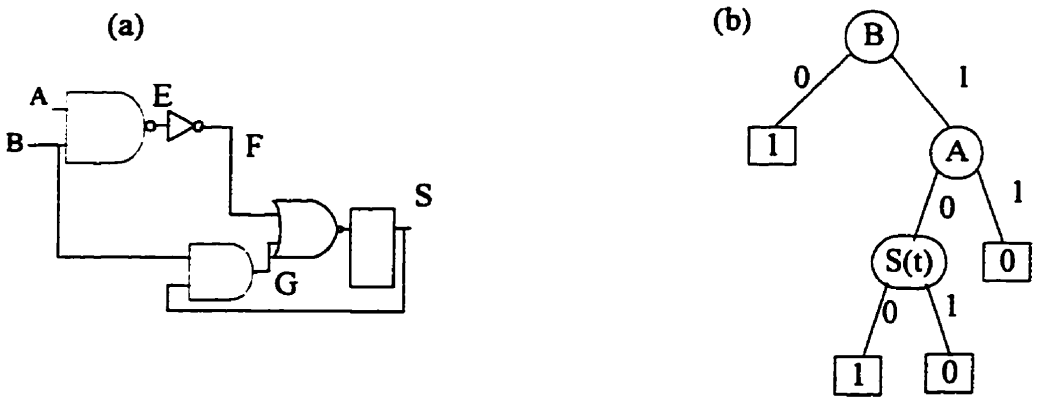


Figure 3.6 Example of (a) a sequential loop and (b) the BDD representation of its output s(t+1)

Modeling accurately the spatial and the temporal dependencies is very time consuming in general. Simplifying assumptions are usually considered because they offer good trade-offs between computational complexity and accuracy. Design Power [130], for instance, assumes a zero delay model in probabilistic-based mode. In the present paper, the mobility measure [124], which is based on transition probabilities, is used for estimating the power consumption. This measure improves the modeling of temporal correlations in sequential circuits as compared with classical controllabilities. The spatial correlation, which is usually the most demanding characteristic to model, is neglected here. We will show that the independence assumption (no spatial correlation) offers a good trade-off between estimation accuracy and its computational complexity, and this is particularly true for large circuits. Moreover, we assume here that glitching is not a major contributor to the power consumption. Therefore, we consider a zero delay model. The next section reviews the main prior contributions to power estimation in sequential circuits directly relevant to this work.

3.5.2 Literature Review

In sequential circuits, using purely combinational logic estimates may result in significant inaccuracies [89]. Indeed, steady-state state probabilities are needed for accurately estimating the power consumption. However, it is widely known that the exact calculation of the steady-state state probabilities is a time-consuming process. The well-known Chapman-Kolmogorov equations (see Appendix 7), for instance, require the solution of a set of 2^n linear equations, where "n" is the number of memory elements in the circuit. Furthermore, it is extremely difficult to extract the state transition informations from the structure of large sequential machines. These informations are needed to build the set of 2^n linear equations mentioned above.

In order to avoid the computational complexity of obtaining the steady-state state probabilities, Gho et al [51] assume that the probability of being in any of the 2^n states of a sequential machine is uniform. They suggest that this assumption may be valid in sequential circuits with strongly connected state transition graphs (STGs). However, as will be shown in this paper, this assumption is only a very rough approximation of reality. Monteiro et al [89] and Tsui et al. [138] have independently proposed similar approximate approaches for estimating the steady-state state probabilities. Their methods are several orders of magnitude faster than exact steady-state state probability calculation methods. With their methods, the steady-state line probabilities of the "n" flip-flop (FF) outputs are estimated instead of the steady-state state probabilities. Their methods require the solution of a nonlinear system of equations of size "n", extracted from the combinational logic implementing the next state function of the sequential circuit, where the "n" variables correspond to steady-state line probabilities. Then, these line probabilities, together with those of the primary inputs, are fed to a tool for estimating the average power consump-

tion in the combinational logic part. In Figure 3.7, a sketch of the approximate methods proposed in [89] and [138] is depicted. This sketch corresponds to the basic framework for several power estimation methods proposed in the literature. In fact, estimating power consumption in sequential circuits is a two-step process. In the first step, study-state line or state probabilities in sequential elements are estimated by considering the circuit as a global Finite State Machine (FSM). In the second phase, the combinational parts are considered taking into account the results of the steady-state line (state) probabilities computed in the first step [95][137].

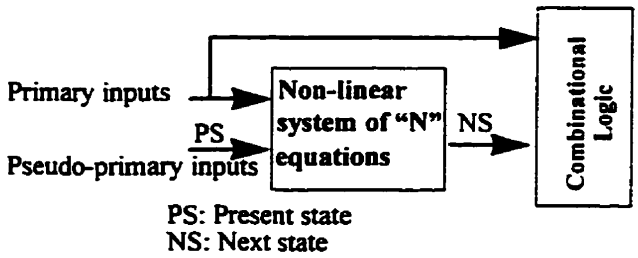


Figure 3.7 A sketch of the methods proposed by Monteiro et al. and Tsui et al.

In this paper, an efficient framework for computing the transition probabilities of large sequential circuits is proposed. This framework improves the state-of-the-art in several ways. The computation of the transition probabilities for sequential circuits is performed with a method similar to the one used with combinational circuits. The proposed method does not use a two-step process. It avoids using state transition graphs or extraction of the next steady-state line functions for building the nonlinear system of equations, as mentioned above. The propagation of the transition probabilities is performed iteratively using simple formulas expressed at the gate level. With the proposed method, estimating the average power consumption for large combinational as well as sequential circuits, such as the s38584 ISCAS 89 benchmark circuit, is achieved in a reasonable time with an acceptable accuracy.

In section 3.5.3, we describe a framework for computing transition probabilities of large sequential circuits. In section 3.5.4, an effective approximate transition probability calculation procedure is proposed for efficiently estimating transition probabilities in large sequential circuits. In section 3.5.5, simulation results are reported to demonstrate the effectiveness of the proposed method. Our main conclusions and remarks are summarized in section 3.5.6.

3.5.3 Computing Transition Probabilities With an Iterative Method

In this section an iterative method for computing the transition probabilities is considered. The transition probabilities, known here as mobilities, are iteratively computed for both the memory elements and the internal nodes, as shown in Figure 3.8. At each iteration, and until stabilization is reached, the transition probabilities of "n" state bits, estimated in the previous iteration, together with the primary input probabilities, are fed to the procedure for computing the next state probabilities.

In this context, where an iterative procedure is used for computing the steady-state line probabilities, the number of iterations needed to reach stabilization is a key parameter determining the computational complexity. Moreover, the existence of asymptotic stabilization should be guaranteed to ensure reliable results.

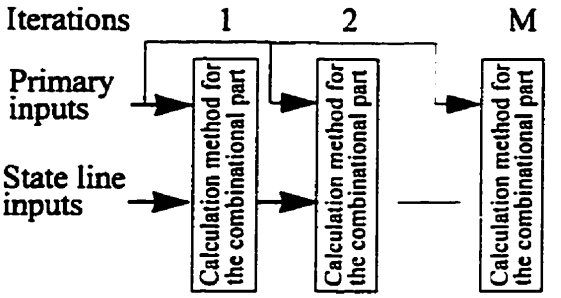


Figure 3.8 An iterative model for computing the mobilities in sequential circuits

In a previous paper, Soufi et al [129] showed that signal probabilities of sequential circuits, starting from a completely unknown state, usually reach stabilization after only few iterations. The convergence to stable values is guaranteed for "initializable circuits". It was shown in [129] that if the circuit is initializable, after some iterations, the circuit forgets its initial state. For hard-to-initialize circuits, the stabilization is not always guaranteed, but it is possible in several cases [129]. For instance, the signal probabilities of the ISCAS 89 s38417 circuit reach stabilization after only 50 iterations. A node is declared stabilized if the differences between the probability values at the iterations "t" and "t-1" are lower than a given threshold set arbitrarily to ".01". Moreover, in non-initializable circuits, the sum of the probability to 1 and the probability to 0 at some nodes does not reach 1, and in some cases, that sum which the node initialization probability remains stuck at 0, indicating that initialization may be impossible.

In the present paper, we are interested in estimating power consumption instead of computing initializablity of sequential circuits as in [129]. Therefore, assuming a completely unknown starting state is no more required, particularly for non initializable circuits where an unknown starting state may result in signal probabilities of some nodes stuck at 0 as indicated above. In non initializable circuits, the steady-state probability of each state is not always unique. It may depend on the initial state. For instance, in the circuit depicted in Figure 3.9, where the majority of 1s or 0s is computed and fed back to the register, the steady-state state probabilities depend on the starting state. Starting from state "101"

leads, in Figure 3.9, to the absorbent state "111", however, starting from state "100" leads to the state "000".

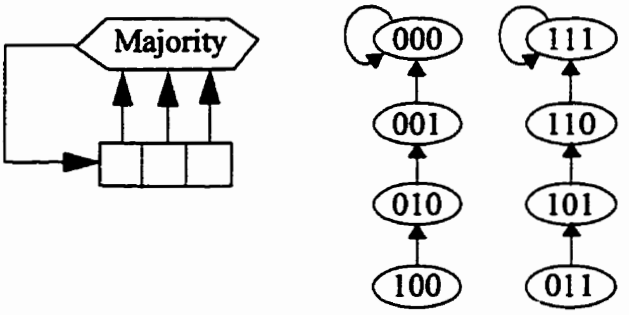


Figure 3.9 Example of non-initializable circuit

For the circuit of Figure 3.9, which is not initializable, using the framework of Figure 3.8 and the assumption that at time 0 we start from any state with equal probability, leads to .5 probability to be in state "111" and .5 probability to be in state "000". This means that, if we repeat the experiment several times, on average, we should be with .5 probability in state "111" and with .5 probability in state "000". The behavior described above can be found in several autonomous state machines, such as LFSRs and Cellular Automata-based generators without maximal cycle length. In these circuits, the state space is fragmented in independent cycles and no starting state allows covering all 2^d states, even though they are all reachable by changing the initial conditions.

In the following, we assume that at time 0 we start from any state with equal probability. The stabilization to the true probabilities is guaranteed for initializable circuits as discussed above. For non-initializable circuits, stabilization may not always be guaranteed, but since the starting state is random, stabilization for circuits such that depicted in Figure 3.9 is possible. Experiments conducted on the ISCAS 89 benchmarks shows that all non-initializable circuits reach stabilization after a reasonable number of iterations, provided

that the initial state is random. In Figure 3.10, starting from a random initial state, the dynamics of the number of FFs where stabilization has not been reached, as a function of the number of iterations, are depicted for a large subset of the well known sequential ISCAS 89 benchmarks.

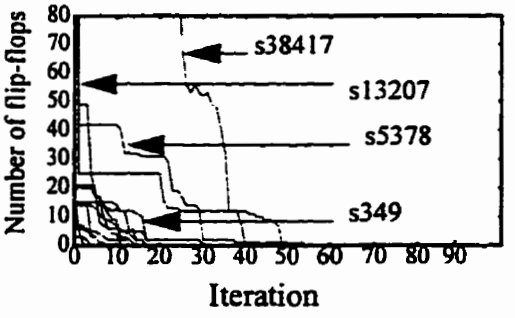


Figure 3.10 The number of flip-flops where stabilization has been reached as a function of the number of steps.

In conclusion, the framework of Figure 3.8 can handle a large sequential circuits, including non-ergodic circuits (such as the one in Figure 3.9) not considered in the literature. A random starting state permits to compute an average of the power dissipation in the case of initializable circuits. However, in the case of non-initializable circuits, the method produces an average of all average power dissipations computed for all possible starting states.

Means of computing transition probabilities for combinational and sequential circuits, that we call mobilities, are briefly reviewed below. Then, a simple procedure based on the framework of Figure 3.8 for propagating these measures in sequential circuits is presented.

3.5.4 Mobility Measures

Before going further, let us show through an example how calculation methods for combinational circuits could be used within the framework of Figure 3.8. Let us consider the example depicted in Figure 3.6. In general, the probability that a rising transition occurs between time "t" and "t+1" is given by

$$P_{0->1}^{t->(t+1)}(S) = P_0^t(S) \times P_1^{t+1}(S)$$

where $P_0^t(S)$ is the probability of "S" to be "0" at time "t" and $P_1^{t+1}(S)$ is the probability of "S" to be "1" at time "t+1".

Using the BDD representation from Figure 3.6, we have

$$\begin{aligned} P_1^{t+1}(s) &= P_0^{t+1}(B) + (P_1^{t+1}(B) \times P_0^{t+1}(A) \times P_0^t(S)) \\ P_0^t(s) &= (P_1^t(B) \times P_1^t(A)) + (P_1^t(B) \times P_0^t(A) \times P_1^{t-1}(S)) \end{aligned} \tag{3.8}$$

Assuming that at time $t = 0$ we can start with equal probabilities from any state, we have

$$P_0^0(s) = P_1^0(s) = \frac{1}{2}$$

Using equation Equation 3.8, at time $t = 1$, we should have

$$P_0^t(S) = \frac{3}{8}$$

$$P_1^t(S) = \frac{5}{8}$$

therefore,

$$P_{0->1}^{0->1}(S) = P_0^0(S) \times P_1^1(S) = \frac{5}{16}$$

This calculation can be repeated until stabilization, and asymptotically, we find

$$P_0(S) = \frac{2}{5}$$

$$P_1(S) = \frac{3}{5}$$

thus,

$$P_{0->1}(S) = P_0(S) \times P_1(S) = \frac{6}{25}$$

Figure 3.11 shows the dynamic evolution of the predicted static probabilities $P_1^t(S)$, $P_0^t(S)$ and the transition probability $P_{0->1}^t(S)$ as a function of time "t".

It remarkable that using a random starting state instead of a known X state as in [128] leads to state probabilities that increase or decrease, thus the monotonical increase of probabilities demonstrated in the framework of [128] is no longer verified.

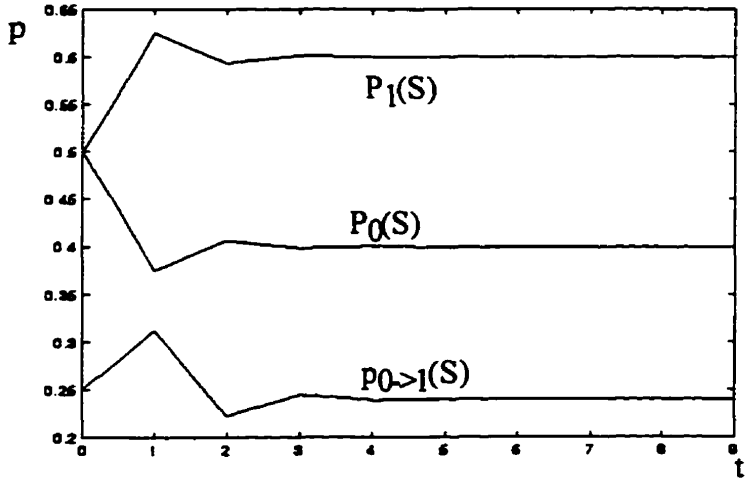


Figure 3.11 Dynamic of static and transition probabilities of "S" as a function of time "t"

In the following, we will use the proposed framework with an approximate probability calculation method. The approximation made here is similar to the one widely made for computing the signal probabilities of digital circuits, where independence between the inputs of each gate is assumed. This is necessary for efficiently computing the signal tran-

sition probabilities in large sequential circuits. The benchmark s38417, with 1636 FFs for instance, would not be manageable with the methods based on state enumeration. Furthermore, we will show that, very often, the average inaccuracies introduced by these simplifying assumptions are not very large.

Mobilities are defined as the probability that a transition will occur on a line "l" in a given circuit. This transition can be up (0 -> 1), down (1 -> 0), or between identical states (0 -> 0 and 1->1). We denote these mobilities $m_{01}^{l}(t)$, $m_{10}^{l}(t)$, $m_{00}^{l}(t)$, $m_{11}^{l}(t)$, where, for instance, $m_{01}^{l}(t)$ is defined as:

$$m_{01}^{l}(t) = \text{probability that } l(t) = 0 \text{ and } l(t+1) = 1$$

The other mobilities follow similarly.

In combinational circuits, the two events

$$l(t) = 0 \qquad \text{and} \qquad l(t+1) = 1$$

are mutually independent, and of course, they are time-independent if their corresponding input vectors are independent. Then, we can obtain

$$m_{01}^{l}(t) = m_{01}^{l} = c_0(l) \times c_1(l) \tag{3.9}$$

where, $c_0(l)$ and $c_1(l)$ are the line probabilities of "l".

In combinational circuits, the mobilities can be directly computed from the steady line probabilities. Therefore, these probabilities are sufficient to characterize the average power consumption.

In sequential circuits, the two above mentioned events

$$l(t) = 0 \qquad \text{and} \qquad l(t+1) = 1$$

are neither mutually independent nor time-independent. As a consequence, equation Equation 3.9 is no longer valid. Using Markov chain processes, it is possible to compute the exact mobilities for every FF output. For instance, note that each transition (0 -> 1) on a flip-flop output corresponds to a transition from the half state space "sub_0", where this flip-flop output is at "0", to the other half "sub_1", where the flip-flop output is at "1". In general, we obtain

$$m_{xy}(l) = \sum_{\left(j \in sub_y\right)} \sum_{\left(i \in sub_x\right)} \pi_i P_{ij} \tag{3.10}$$

with $0 \le m_{xy} \le 1$ and $(x,y) \in \{0,1\}^2$.

In the following, we derive formulas at the gate level similar to those already proposed for the computation of steady line signal probabilities. These formulas allow propagation of mobility values in a way very similar to how approximate controllabilities are computed [54]. Note however that for tractability reasons, independence between internal nodes is assumed.

3.5.4.1 Propagation of Mobility Signals

As discussed above, the computation of mobilities directly from the Markov chain is a time-consuming process and Markov chains proved more useful in the demonstration of general properties [129]. In order to reduce the cost of computing mobilities we now present a procedure exploiting the structure of the circuits, as well as simple formulas defined for each gate type. This procedure is linear with respect to the number of gates in the circuit, times the number of iterations needed to reach stabilization.

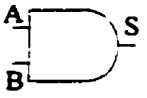


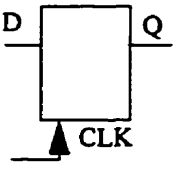
Table 3.6 Transition table of a two input AND gate

A	B	S	A	B	S
0->0	0->0	0->0	1->0	0->0	0->0
0->0	0->1	0->0	1->0	0->1	0->0
0->0	1->0	0->0	1->0	1->0	1->0
0->0	1->1	0->0	1->0	1->1	1->0
0->1	0->0	0->0	1->1	0->0	0->0
0->1	0->1	0->1	1->1	0->1	0->1
0->1	1->0	0->0	1->1	1->0	1->0
0->1	1->1	0->1	1->1	1->1	1->1

Let us now compute the mobilities in the case of a 2-input AND gate. Table 3.6 lists all possible transitions in a 2-input AND gate. From this table we find that

$$m_{11}(S) = m_{11}(A) \times m_{11}(B)$$
$$m_{01}(S) = m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{11}(B) + m_{11}(A) \times m_{01}(B)$$
$$m_{10}(S) = m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B)$$
$$m_{00}(S) = 1 - (m_{11}(S) + m_{01}(S) + m_{10}(S))$$

For sequential elements, such as D-flip-flops, the mobilities are time-dependent, and they are given by:



$$m_{01}^{Q}(t+1) = m_{01}^{D}(t)$$

Similar formulas may be derived for all gates (see Appendix 6).

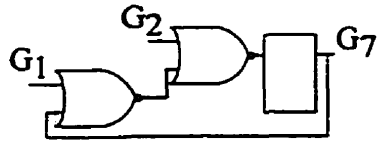


Figure 3.12 A sequential circuit taken from the ISCAS s27 circuit

Figure 3.12 shows an example of a sequential loop extracted from the ISCAS s27 circuit. The mobilities of all nodes may be computed using a simple iterative algorithm. Initially, we assign ".25" to all mobilities of the feedback node "G7":

$$m_{00}^{G7} = m_{01}^{G7} = m_{10}^{G7} = m_{11}^{G7} = \frac{1}{4}$$

We also assume equiprobable input transitions in this example but the method is easily generalized to weighted input vectors. Therefore, the mobilities of the inputs are equal to .25. Then, using formulas such as those proposed for a 2-input AND gate, the mobilities of all nodes are iteratively computed, until their values stabilize within some predefined acceptable accuracy (".01" in this paper).

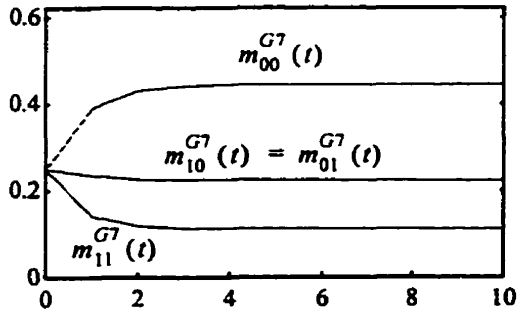


Figure 3.13 Curve describing the dynamic behavior of the mobilities of the node G7 in Equation 3.12

Figure 3.13 shows the dynamic behavior of the mobilities on node "G7" in Figure 3.12. As mentioned before, the mobilities stabilize after few iterations (4 in this case).

3.5.4.2 Relationship between mobilities

In the following, we show that only two of the 4 defined mobilities ($m_{01}^{l}(t)$, $m_{10}^{l}(t)$, $m_{00}^{l}(t)$, $m_{11}^{l}(t)$) need to be computed from the circuit. The other two can be derived from a system of 2 linear equations.

Let us demonstrate the following lemma

Lemma 1:

$$m_{01}^{l}(t) = m_{10}^{l}(t) \tag{3.11}$$

Proof: we know that

$$m_{01}^{l}(t) + m_{10}^{l}(t) + m_{00}^{l}(t) + m_{11}^{l}(t) = 1 \tag{3.12}$$

Let us assume without loss of generality that the set of states where $l = 0$ corresponds to the half state space from state 1 to state 2^{n-1}, as a consequence, we can write

$$m_{01}^{l} = \sum_{\substack{i = 1 \to 2^{n-1} \\ j = (2^{n-1}+1) \to 2^{n}}} \pi_i \times p_{ij} \tag{3.13}$$

$$m_{10}^{l} = \sum_{\substack{i = 1 \to 2^{n-1} \\ j = (2^{n-1}+1) \to 2^{n}}} \pi_j \times p_{ji} \tag{3.14}$$

$$m_{00}^{l} = \sum_{\substack{i = 1 \to 2^{n-1} \\ j = 1 \to 2^{n-1}}} \pi_i \times p_{ij} \tag{3.15}$$

By adding equations (3.13) and (3.15), we find

$$m_{01}^{l} + m_{00}^{l} = \sum_{\substack{i = 1 \to 2^{n-1} \\ j = (2^{n-1}+1) \to 2^{n}}} \pi_i \times p_{ij} + \sum_{\substack{i = 1 \to 2^{n-1} \\ j = 1 \to 2^{n-1}}} \pi_i \times p_{ij} \qquad (3.16)$$

Note that the index "j" in equation (3.16) varies from 1 to 2^{n}, as a consequence we can write

$$m_{01}^{l} + m_{00}^{l} = \sum_{\substack{i = 1 \to 2^{n-1} \\ j = 1 \to 2^{n}}} \pi_i \times p_{ij}$$

then,

$$\begin{aligned} m_{01}^{l} + m_{00}^{l} = {} & \pi_1 \times \left(p_{11} + p_{12} + \ldots\ldots + p_{12^{n}} \right) + \\ & \pi_2 \times \left(p_{21} + p_{22} + \ldots\ldots + p_{22^{n}} \right) + \\ & \ldots\ldots\ldots\ldots \\ & \pi_{2^{n-1}} \times \left(p_{2^{n-1}1} + p_{2^{n-1}2} + \ldots\ldots + p_{2^{n-1}2^{n}} \right) \end{aligned}$$

the summations between brackets are, by definition of Markov chain processes, always 1. As a consequence, we can write

$$m_{01}^{l} + m_{00}^{l} = \sum_{i = 1 \to 2^{n-1}} \pi_i$$

but, by definition (see [124])

$$\text{probability that } l \text{ is } 0 = C_0 = m_{10}^{l} + m_{00}^{l} = \sum_{i = 1 \to 2^{n-1}} \pi_i \qquad (3.17)$$

that is, the probability to be at state 0, which can be computed by summing the state probabilities over the half state space where $l = 0$, is the sum of the probabilities to go to state 0.

From equations (3.16) and (3.17), we find,

$$m_{10}^{l} + m_{00}^{l} = m_{01}^{l} + m_{00}^{l}$$

By eliminating m_{00}^{l} from both sides of equation (3.17), we find,

$$m_{10}^{l} = m_{01}^{l} \qquad \text{Q. E. D}$$

The 4 mobilities can be computed from equations (3.11) and (3.12) if any 2 of the mobilities are first computed from the circuit.

3.5.5 Experimental Results

In order to demonstrate the effectiveness of the transition probability calculation method proposed in this paper, we carried out several experiments using a large subset of the well-known MCNC and ISCAS 89 benchmarks.

Table 3.7 Comparison of run times between our method and other methods proposed in literature

Circuit name	FF	P_{ave}	T(Secs)	#Iterations	T [138] (secs) (1 iteration)	T [89] (secs) (1 iteration)
bbara	4	1117.90	0.133	55	-	-
bbtas	3	374.87	0.017	19	-	-
dk14	3	2136.15	0.067	12	2.71	-
dk15	2	921.92	0.017	10	1.63	-
dk16	5	5115.27	0.167	13	8.43	6
dk17	3	872.21	0.067	25	1.55	-
dk27	3	352.16	0.033	24	-	-
mc	2	393.32	0.033	36	-	-
planet	6	14097.69	0.5	17	15.51	18
s1	5	8016.19	0.2	13	-	-

Table 3.7 Comparison of run times between our method and other methods proposed in literature

Circuit name	FF	P_{ave}	T(Secs)	#Iterations	T [138] (secs) (1 iteration)	T [89] (secs) (1 iteration)
shiftreg	3	145.42	0.033	47	-	0
tav	2	324.08	0.033	34	-	-
tbk	5	14848.86	0.567	19	60.34	46
s27	3	102.62	0.033	7	-	-
s208	8	301.92	0.017	12	-	-
s298	14	797.79	0.067	24	-	-
s344	15	1057.95	0.067	20	-	-
s349	15	1063.83	0.05	21	-	-
s382	21	623.23	0.10	26	5.16	
s386	6	1050.57	0.083	23	-	-
s420	16	501.93	0.05	12	-	-
s444	21	679.33	0.10	23	-	-
s510	6	1801.87	0.25	43	7.95	
s526	21	824.92	0.13	28	-	-
s526n	21	816.49	0.15	28	-	-
s641	19	2065.34	0.2	37	-	-
s713	19	2505.21	0.3	47	-	-
s820	5	2428.37	0.18	21	16.06	
s832	5	2452.90	0.167	20	-	-
s838	32	867.10	0.1	12	-	-
s953	29	2406.12	0.317	35	-	25
s1196	18	4132.32	0.033	5	-	82
s1423	74	3695.49	0.35	26	-	289
s1488	6	5163.31	0.217	15	35.23	-
s1494	6	5187.33	0.20	14	-	-
s5378*	179	11266.13	11.20	100(.7%)	-	-
s9234*	228	6460.88	18.28	100(1.5%)	-	-
s13207*	669	11655.11	25.45	100(6%)	-	-
s15850*	597	16215.93	31.3	100(0.2%)	-	-
s35932	1728	118842.78	4.55	12	-	-
s38417*	1636	112516.53	76.8	100(14%)	-	-
s38584*	1452	108409.04	81.88	100(.3%)	-	-

Table 3.7 contains run times and number of iterations for stabilization for a large set of MCNC and ISCAS 89 benchmarks. Stabilization is reached when the difference between each individual mobility computed at cycle "n" and "n-1" is inferior to a threshold set arbitrary to ".001". Let us mention here that for some cases, marked by *, the stabilization according to this criteria was not reached at 100 cycles. In this case, we have reported the percentage of nodes which were still not stabilized. As expected, the run time

is reasonable even for large circuits. For instance, the run time needed for estimating the average power consumption of the s35932 is only 4.55 seconds. These results were obtained using a SPARC 5 (555-85) workstation. A comparison between the run time of the method proposed in this paper and those reported in the literature confirms that our method is less complex. For instance, in the case of tbk, the computation time needed for 1 iteration by the method proposed in [138] is 60.34 seconds on a Sparc 2 station. In the case of the method proposed in [89], a time of 46 seconds on a DEC AXP 3000/500 is reported. No information on the number of iterations was given in [89]. However, only 0.567 second for 19 iterations is needed with our method. The number of iterations required to stabilize with all three methods should be comparable.

The average power consumption measurement "P_{ave}" was estimated based on the following well known model [97]:

$$P_{ave} = 0{,}5 \times \frac{V_{dd}^2}{T_{cycle}} \times \sum_{\text{all node i}} C_i \times E_i \tag{3.18}$$

where C_i is the load capacitance on node "i" and E_i the sum of the mobilities m_{01}^i, m_{10}^i of node "i". C_i was approximated by the number of fan-outs on the node "i". We can rewrite (3.18) as

$$P_{ave} = 0.5 \times \frac{V_{dd}^2}{T_{cycle}} \times \sum_{\text{all node i}} C_i \times \left((m_{01})_i + (m_{10})_i \right) \tag{3.19}$$

Using lemma 1, equation (3.19) can be rewritten as follows

$$P_{ave} = \frac{V_{dd}^2}{T_{cycle}} \times \sum_{\text{all node i}} C_i \times (m_{01})_i \tag{3.20}$$

Table 3.8 Errors introduced by the approximate method proposed in this paper

Circuit	No. of states	Steady-state states probabilities			|err| %
		Min	Ave	Max	
bbara	10	.002	.1	.267	6.5
bbtas	6	.078	.167	.235	2.6
dk14	7	.016	.143	.381	9.3
dk15	4	.041	.25	.381	7.5
dk16	27	.004	.037	.075	12.4
dk17	8	.009	.125	.25	8.6
dk27	7	.048	.143	.214	2.6
mc	4	.143	.25	.428	9
planet	48	.002	.021	.054	4.1
s1	20	.010	.05	.119	1.2
shiftreg	8	.125	.125	.125	5.6
tav	4	.25	.25	.25	3
tbk	32	.005	.031	.341	1.6

In Table 3.8, statistics based on the exact steady-state probabilities are reported. For each circuit, we reported the number of states, the minimum, the average and the maximum of the steady-state states probabilities computed using the exact well-known Chapman-Kolmogorov equations [99]. We also reported the average of the absolute values of the errors produced by the approximate method proposed in this paper. These errors were obtained by adding the absolute values of the errors of all steady-state line bits, dividing by the number of line bits, and multiplying the results by 100.

Note the large differences between the steady-state state probabilities. For instance, in the case of "dk 15", the minimum steady-state state probability is .041, the maximum is .381 and on average, the steady-state state probability is .25. Furthermore, Table 3.8 shows that average absolute values of the errors do not exceed 12.4% and on average |err| is 5.69%. However, the errors introduced by our method for estimating the mobilities of all nodes can be positive or negative, and for large circuits, the errors may be mutually compensated. For instance, in the case of the "planet" benchmark, we find that the sum of

all errors is null. Nevertheless, it was found that for the circuits of Table 3.8, which are all fairly small, the errors are often of the same sign.

In order to model the effect of circuit size, let us estimate the relative error on the average power consumption in the special case where the nodes are assumed independent. The average power consumption is given by equation (3.20), and consequently we can write its expected value as

$$Exp\left[P_{ave}\right] = Exp\left[A \times \sum_{\text{all node i}} C_i \times E_i\right]$$

where $A = 0{,}5 \times \frac{V_{dd}^2}{T_{cycle}}$.

Therefore, we find

$$Exp\left[P_{ave}\right] = A \times \sum_{\text{all node i}} C_i \times Exp\left[E_i\right] \tag{3.21}$$

On the other hand, we have

$$var\left[P_{ave}\right] = A^2 \times var\left[\sum_{\text{all node i}} C_i \times E_i\right] \tag{3.22}$$

If the random variables E_i are independent and all $C_i = 1$, equations (3.21) and (3.22) can be rewritten as

$$Exp\left[P_{ave}\right] = A \times N \times \overline{Exp\left[E_i\right]} \tag{3.23}$$

and

$$var\left[P_{ave}\right] = A^2 \times N \times \overline{var\left[E_i\right]} \tag{3.24}$$

where N is the total number of nodes in the circuit and the line above the variables designates the average value of that variable over all nodes. Therefore, the relative error can be written as

$$\frac{\Delta P_{ave}}{Exp\left[P_{ave}\right]} \approx \frac{\sqrt{var\left[P_{ave}\right]}}{Exp\left[P_{ave}\right]} \tag{3.25}$$

$$\frac{\Delta P_{ave}}{Exp\left[P_{ave}\right]} \approx \frac{\sqrt{\overline{var\left[E_i\right]}}}{Exp\left[E_i\right]} \times \frac{1}{\sqrt{N}} \tag{3.26}$$

This is a classic property expected when summing independent random variables [99]. From equation (3.26), we can observe that the relative errors decrease as the square root of the total number of gates in the circuits. As a consequence, the approximate method proposed here tends to become more accurate when the number of nodes increases.

3.5.6 Conclusion

We have presented an efficient method for estimating the average power consumption in large sequential circuits. This framework is based on an iterative process which can be integrated with several accurate computation methods proposed in the literature for combinational circuits. We also proposed an approximate computation method for estimating the switching activity. This method is typically two orders of magnitude less complex than previously reported methods. Indeed, its complexity is linear with respect to the number of gates times a limited number of iterations (a maximum of 100 cycles). The procedure is based on simple formulas defined at the gate level for each gate type. These formulas are similar to those already proposed for computing signal probabilities [54]. Finally, it was

shown that the errors introduced by this procedure are not very large. Moreover, the accuracy of the predicted average power should improve with the size of the circuit.

CHAPITRE 4

Une méthodologie efficace pour l'insertion des points de test dans les circuits séquentiels

4.1 Résumé

Dans le présent chapitre, qui a fait l'object d'une publication soumise à "IEEE Transactions on Computer-Aided Design", une nouvelle méthode d'insertion des points de test dans les circuits séquentiels est proposée. Cette méthode est basée sur les notions de mesures de mobilité développées au chapitre 3. Nous avons déjà montré l'efficacité des mesures de mobilité relativement au problème de la quantification des problèmes de test dans le cas des circuits séquentiels. Dans ce chapitre, nous montrerons comment utiliser les mesures de mobilité pour guider le processus d'insertion des points de test de façon efficace. De plus, plusieurs nouvelles notions reliées au problème de l'insertion des points test seront étudiées et des solutions heuristiques pratiques y seront proposées. Ainsi, nous montrerons que les quatre mesures de mobilité définies au chapitre 3 sont d'importance égale. Un problème de test est probable si une d'entre elles est faible. Une mobilité est déclarée faible si sa valeur est inférieure à un seuil de détection des problèmes de mobilité MTh défini un peu plus bas. En utilisant les mesures de mobilité, nous définirons trois types de problèmes de contrôle:

1. quand $m_{00}(t) < MTh$, nous avons un problème de transition de 0 -> 0,

2. quand $m_{11}(t) < MTh$, nous avons un problème de transition de 1-> 1,
3. quand $(m_{01}(t) = m_{10}(t)) < MTh$, nous avons un problème d'activité.

À chaque type de problème de mobilité, nous proposerons une structure de point de test adéquate. En effet, une porte AND, par exemple (voir section 4.2.3), peut générer des effets négatifs (des pertes de couverture) dans certaines situations, lorsqu'elle est utilisée pour régler un problème de transition 0->0. Par conséquent, une porte AND, classiquement utilisée pour régler un problème de contrôle à 0, n'est parfois pas très bien adaptée aux problèmes de transitions 0->0. Une procédure heuristique pour sélectionner une structure de point de test adéquate pour chaque type de problème de mobilité est présentée à la section 4.2.3.

L'insertion simultanée de plusieurs points de test peut avoir des effets positifs comme elle peut avoir des effets négatifs. En effet, dans certaines situations, un problème de test peut nécessiter plusieurs points de test insérés simultanément pour être détecté. L'insertion par groupe dans ce cas peut éliminer la nécessité de dédier un point de test au problème en question et ainsi produire de meilleurs résultats. Cependant, avec l'insertion par groupe, nous devons faire attention aux situations de conflit où les effets cumulatifs de plusieurs points de test peuvent résulter en des pertes de couverture de pannes. Par exemple, dans le cas du circuit de la figure 4.1, qui est largement mentionné dans la littérature comme un cas résistant au test pseudo-aléatoire pondéré, le test de A est incompatible avec le test de B. Dans ce chapitre, nous proposons une procédure de test multiphase où chaque phase ne

contient que des points de test compatibles. Ces phases sont activées séquentiellement l'une après l'autre.

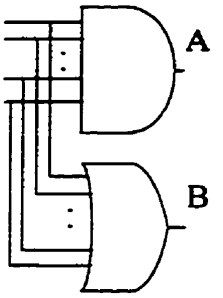


Figure 4.1 Un exemple d'un circuit résistant à un test pondéré avec un seul ensemble de poids.

Pour valider les différentes heuristiques d'insertion de points de test proposées dans ce chapitre, nous avons fait plusieurs expériences sur un grand ensemble de circuits d'essai séquentiels. Les résultats expérimentaux nous ont montré l'efficacité de la méthode proposée. Ainsi, avec quelques points de test répartis sur un nombre limité de phases, nous avons nettement amélioré la qualité du test de ces circuits.

4.2 A Methodology for Efficiently Inserting Test Points in Sequential Circuits for Pseudo-Random Testing

M. Soufi, Y. Savaria and B. Kaminska

Ecole Polytechnique de Montréal

Department of Electrical and Computer Engineering

P.O Box 6079, Station Centre-ville

Montréal, PQ, H3C 3A7

Email: savaria@vlsi.polymtl.ca

Tel: (514) 340-4737

Abstract

In this paper, a new multiphase test point insertion methodology for sequential circuits is presented. This methodology is based on a new testability measure called mobility that better quantifies testing problems in sequential circuits. Moreover, new notions related to test point insertion will be studied. Each test point may be associated with positive and negative side effects. These effects, as well as the impact of the joint insertion of several test points, are discussed and heuristic solutions to handle them are proposed. Results obtained by applying these heuristics to a large set of sequential benchmark circuits are reported.

4.2.1 Introduction

Design For Testability (DFT) comprises well established testing methodologies which are used to keep testing costs within reasonable bounds [4]. In fact, the complexity of modern digital circuits is rapidly increasing, and the trend towards more complex systems should continue for some time [22]. Unfortunately, this increase in complexity is

associated with limited accessibility to internal nodes, which is a bottleneck faced by test engineers.

Most DFT approaches require extra hardware and circuit modifications. Each modification is associated with extra overheads that may impact different aspects of the design. Area overhead, timing penalties, extra input/output ports and additional power dissipation are examples of these overheads. These modifications aim to increase accessibility and thus lower the testing efforts. For instance, the well known scan techniques chain together several (or all) flip-flops in one (or several) long shift register. The chain is accessed by an extra input and its content is read through an extra output. Scan techniques are very effective at reducing test generation efforts and enhancing test quality.

In many popular scan techniques, the test vectors are not applied in parallel at the operational speed of the circuit. Rather long vectors are usually scanned in at a relatively slow but feasible speed. Then, the circuit is exercised by a single or a small set of fast clock pulses [6]. This method has shortcomings, particularly in high speed applications [108][125]. In fact, in high speed applications, high performance clock distribution networks are generally used. These networks are not designed to propagate isolated pulses, and such pulses would be subject to significant distortions, affecting the performance of these circuits. Moreover, it is very well known that applying parallel test vectors, at the operational speed of the circuit, is more effective at identifying defective circuits than applying the same test in a scan mode [83]. This is related to the fact that not all defects are accurately modeled by the widely used stuck-at fault model.

Another approach, which is gaining more credibility in the design community, is Built-In Self-Test (BIST) [83]. Indeed, BIST has several desirable characteristics. It offers

the possibility to self test the chip internally, and hence avoids going through the input and output ports, which is very important for high performance applications [125]. The complexity of external test equipments is reduced by moving some or all of their functions inside the chip itself. Moreover, with BIST, we have the possibility of reusing tests developed at lower level (component or block levels) for higher level tests (card, system levels).

Many BIST techniques rely on pseudo-random testing, where the test patterns are generated using efficient pseudo-random test generators, and test responses are evaluated using response analyzers. However, it turns out that in several cases, some faults are difficult to test with pseudo-random vectors. These faults are called resistant. Test point insertion methods were suggested as possible solutions to deal with resistant faults. A formulation of the test point insertion problem consists of selecting a small number of locations for inserting additional hardware, in order to ease test development, and this should be within reasonable bounds for area overhead, timing penalties and number of additional I/O ports required for testing. The four relevant aspects of the test point insertion problems are:

1. the structure of the added hardware,
2. the locations where such hardware should be inserted,
3. the condensation of test points to reduce the additional I/Os required for testing,
4. the complexity and situation of the pseudo-random sources used to feed the test points.

There are two kinds of test points. Control points used to enhance controllability and observation points used to enhance observability. AND and OR gates were suggested as test point structures to enhance 0-controllabilities and 1-controllabilities respectively. For observabilities, additional outputs are used. Recently, Savaria el al. [114] proposed the FO approach for CMOS technology. With this approach, the gate feeding the net where the

insertion should be performed is replaced by an equivalent complex testable gate. XOR gates were also used by several authors for enhancing observabilities [46][63] as well as controllabilities [126]. Another structure called Test-Cell was also suggested as a test point structure in the literature [62]. Test-cell based methodologies are similar to scan techniques, however, test cells can be inserted even in the combinational part of sequential circuits. A test cell is a flip-flop with scan mechanisms. After insertion, test cells are chained together in a long shift register as in scan methods.

Finding an optimal set of locations for test point insertion is an NP-complete problem [73]. Test point insertion methods proposed in the literature may be divided into 2 classes. First, fault-simulation based techniques. Second, testability-measures based techniques. In the first category, we find the work of Briers and Totton [25] as well as that of Iyengar and Brand [63], which are representative. With these methods, fault simulations are first invoked to collect some statistics on the test problems. These statistics are then used to determine a small number of locations for insertion. However, fault-simulation based techniques are expensive and they also have a limited accuracy, since they rely on fault simulation of a limited set of faults. Heuristic methods based on testability measures [119][145] were then proposed to overcome these shortcomings. They have produced interesting results in combinational circuits. Some attempts to generalize these approaches to sequential circuits have recently been proposed [75][90].

Test point condensation is another important aspect of test point insertion methods. In fact, the I/O budget is usually limited. As a consequence, condensation techniques used to limit the number of additional I/Os needed by the test points are always highly desirable. For instance, in [145], the authors proposed to use several modes which are invoked separately. Each mode only includes a small subset of the total number of test points, such

that the additional I/Os required for testing can be shared between different modes. Multiplexor-based techniques have also been used for condensation. With these techniques, the same I/Os are shared between the actual function and the test points.

Pseudo-random sources are usually generated by dedicated hardware (pseudo-random generators). This may negatively impact the performance of the system. The generators are often heavily loaded. The idea of using internal points as pseudo-random sources has been first proposed for partial reset in [126]. In fact, an internal node can be used as a pseudo-random source, provided that its pseudo-random characteristics are good enough, and that the correlation between this node and other test points is limited. Recently, the same idea was successfully used in a test point insertion method for sequential circuits [90].

In the present paper, a new test point insertion method for sequential circuits is proposed. This method is based on a new testability measure called mobility, already introduced in [124]. We also propose in this paper several heuristics based on new important notions related to test point insertion. In section 4.2.2, we will briefly introduce the mobility measure. In section 4.2.3, we will discuss the effects on mobilities of several test point structures proposed in the literature. In section 4.2.4, joint effects of multiple simultaneously active test points is analyzed. Several heuristics for test point insertion will be presented in section 4.2.5. Observability point insertion will be the subject of section 4.2.6. Experimental results are presented in section 4.2.7, and our main conclusions and remarks are summarized in section 4.2.8.

4.2.2 Modeling Fault detection in Sequential Circuits

COP numbers have been successfully used to determine good locations for test points insertion in combinational circuits [119][145]. However, COP numbers are not directly suitable for sequential circuits. Recently, several authors [18][75][129] have proposed probabilistic testability measures for sequential circuits. Nevertheless, all these measures are blind to some important testing problems in sequential circuits, as shown in [124]. In order to overcome these shortcomings, the mobility measure is used. Some basic definitions related to mobilities are repeated here for clarity and convenience. Mobility is based on the concept of transition probability calculations. For each node in the circuit, four mobilities are defined ($m^l_{01}(t)$, $m^l_{10}(t)$, $m^l_{00}(t)$, $m^l_{11}(t)$). They represent all possible transition probabilities. Mobilities are related to controllabilities as follows [124]:

$$\begin{aligned} c^l_0 &= m^l_{00} + m^l_{10} \\ c^l_1 &= m^l_{01} + m^l_{11} \end{aligned} \tag{4.1}$$

The mobilities of the output of each gate can be directly computed from the mobilities of its inputs, using simple formulas similar to those proposed for controllabilities. For instance, in the case of a 2-input AND gate, the mobilities of its output "S" are related to those of its inputs {"A", "B"} as follows:

$$\begin{aligned} m_{11}(S) &= m_{11}(A) \times m_{11}(B) \\ m_{01}(S) &= m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{11}(B) + m_{11}(A) \times m_{01}(B) \\ m_{10}(S) &= m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B) \\ m_{00}(S) &= 1 - (m_{11}(S) + m_{10}(S) + m_{01}(S)) \end{aligned} \tag{4.2}$$

Note that only 2 of the 4 mobilities need to be computed. The other two can be derived from the following system of two linear equations [123]:

$$m_{01}^{l}(t) = m_{10}^{l}(t) \tag{4.3}$$

$$m_{01}^{l}(t) + m_{10}^{l}(t) + m_{00}^{l}(t) + m_{11}^{l}(t) = 1 \tag{4.4}$$

4.2.2.1 Complexity of Computing the Mobilities

The rules for computing the mobilities are slightly more complex than those used for computing controllabilities (see Appendix 8, page 164). A comparison between them shows that 2 additional multiplications and 4 additional additions are required in the computation of the mobilities. As a consequence, the complexity of computing the mobilities for one iteration (TM), as a function of the complexity of computing the controllabilities (TC), also for one iteration, can be written as follows:

$$TM = TC + N(2 \times mult + 4 \times add) \tag{4.5}$$

where N is the total number of gates in the circuit, $mult$ and add are constants representing the times required to compute a single multiplication and a single addition respectively. This expression assumes that only one operation is executed at a time, and it would not be representative of the processing time on modern superscalar machines, where operations could be performed in parallel.

Since we know that TC is linear with respect to N:

$$TC = O(N) \tag{4.6}$$

thus, from equations (4.5) and (4.6), we can write

$$TM = O(N) \tag{4.7}$$

In conclusion, the complexity of computing mobilities is also linear with respect to the total number of gates in the circuit. However, the hidden coefficient in the complexity

polynomial (equation (4.7)) is slightly greater than the one representing controllabilities computation (equation (4.6)).

4.2.2.2 Mobility as a Testability Measure for Sequential Circuits

The superiority of the mobilities in predicting controllability problems was also shown in [124]. In that paper, we showed that good controllabilities may be associated with poor mobilities, which may lead to a testing problem. This situation arises, for instance, in a very popular digital block, the binary counter. Indeed, even if the limiting probabilities are as good as 0,5 for all bits, it is clear that the sequences needed to test these bits are of different lengths. For instance, in a 5-bit counter excited with an equiprobable pseudo-random signal on its enable input, a transition occurs in the least significant bit every two cycles on average, however, in the most significant bit, the average time between two successive transitions of the same polarity is 2^6 cycles.

Let us now recall some analysis of detection probabilities as proposed by Shedletsky and McCluskey [122].

Let us assume that the detection probability (detectability) per pattern for a given stuck-at-0 fault is q_0 (we can analyze stuck-at-1 faults the same way). The cumulative distribution function of the error latency of this fault is:

$$E_c = 1 - (1 - q_0)^N \tag{4.8}$$

The error latency is defined as the number of patterns applied until the fault is detected. Of course, in this model, we assume that the circuit is combinational, the input patterns are independent and the detection problem is a Bernoulli trial process with q_0 as

probability of success. The number of patterns required to reach a given E_c can be easily derived from equation (4.8) as follows:

$$N = \frac{\ln(1 - E_c)}{\ln(1 - q_0)} \tag{4.9}$$

From equation (4.1) and the independence assumption between controllabilities and observability, we can write

$$q_0 = c_1 \times o = (m_{11} + m_{01}) \times o = q_{00} + q_{10} \tag{4.10}$$

where $q_{00} = m_{11} \times o$ and $q_{10} = m_{01} \times o$ are respectively called here 1->1 and 0->1 transition detectabilities.

Equation (4.10) shows that the detectability of a stuck-at-0 fault is the sum of their corresponding 1->1 and 0->1 transitions detectabilities. In sequential circuits, a simple Bernoulli trial model cannot be used to estimate the cumulative distribution function of the error latency, since the input patterns are no more independent. Shedletsky and McCluskey [122] proposed another model for sequential circuits based on Markov chains and the product machine, which is the combination of a fault free and a faulty machine. This model is complicated, and it requires a lot of computational resources for large sequential circuits. Let us assume here that the Bernoulli model is still valid for computing the transition detectabilities in sequential circuits. In this case, we can write

$$\begin{aligned} E_{m_{11}} &= 1 - (1 - q_{00})^N \\ E_{m_{01}} &= 1 - (1 - q_{10})^N \end{aligned} \tag{4.11}$$

where $E_{m_{11}}$ and $E_{m_{01}}$ are the cumulative distribution functions of the 1->1 and 0->1 transition latencies. In the binary counter, for instance, good controllabilities are not

enough and in order to ensure good pseudo-random testability for a stuck-at-0 fault, both cumulative distribution functions of the 1->1 and 0->1 transition latencies should be sufficiently good.

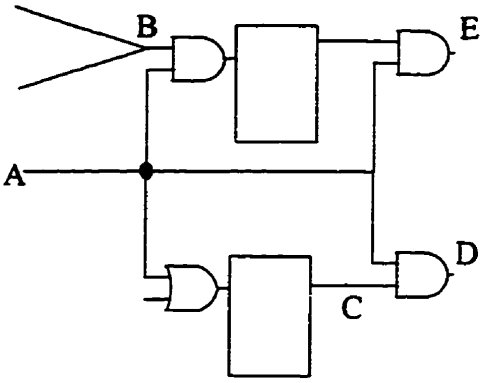


Figure 4.2 Example to show the importance of mobilities for testing

Figure 4.2 shows an interesting situation where low mobilities would reduce testability. For instance, testing "C" stuck-at-1 requires a good 0->1 mobility of "A". If it happens that "A" is fed by the most significant bit of a binary counter, testing "C" stuck-at-1 would be problematic, despite the fact that the controllabilities of "A" in two successive cycles are both 0,5. Another interesting situation is also shown in Figure 4.2, where the 1->1 transition of "A", as reflected by a non zero value of the 1->1 mobility, is necessary to make the input cone of "B" observable. If it happens that "A" is fed by the least significant bit of a counter, the input cone of "B" cannot be observed through E over long time periods. Using the same method used to synthesize the circuit of Figure 4.2, we can build circuits where a large input would only become observable at very infrequent transitions, causing an obvious testability problem.

In conclusion, the 4 mobilities are important. If one of them is too small, testability problems may arise. From the above discussion, we propose to compute the cumulative distribution function of the error latency of a stuck-at-0 fault in sequential circuits with:

$$E_m = min\{E_{m_{11}}, E_{m_{01}}\} \tag{4.12}$$

A similar equation can be written for a stuck-at-1 fault.

One should note here that the model for estimating E_m is a lower bound that reflects the situation where the mobility with the lowest detectability function is necessary to cover the target fault. It is a heuristic model whose usefulness will be confirmed later by experimental results. For instance, in the case of a 5 bit counter, we have depicted the cumulative distribution function of the error latency for a fault on the most significant bit, as computed from equations (4.8) and (4.12). Note that E_c is almost 1 after few iterations, however, E_m is still only 0.8 after 100 cycles.

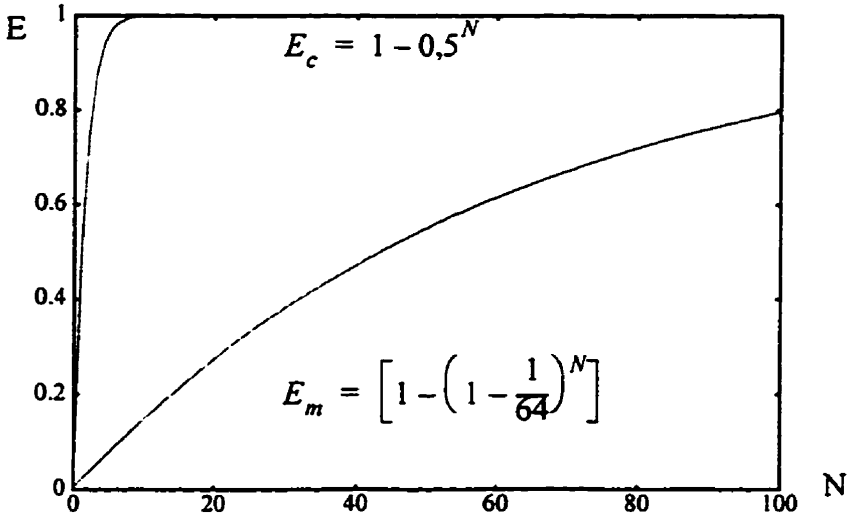


Figure 4.3 Cumulative distribution function of the error latency for a fault on the most significant bit in a 5 bit counter using equations (4.8) and (4.12)

4.2.3 Effect of Test Point Structures on Mobilities:

As mentioned above, COP numbers were widely used in test point insertion methods [119][145]. With these numbers, controllability problems are declared when their corresponding controllabilities are inferior to a given threshold (*CTh*). Similarly, with the mobilities, we propose to use a mobility detection threshold (*MTh*) to detect control problems. Using mobilities, there are 4 types of control related problems. However, since $m^{l}_{01}(t) = m^{l}_{10}(t)$ (equation (4.3)), we can reduce them to 3 instead of 4:

When $m^{l}_{00}(t) < MTh$, we say that a 0->0 transition problem is detected;

when $m^{l}_{11}(t) < MTh$, we say that a 1->1 transition problem is detected; and

when $\left(m^{l}_{10}(t) = m^{l}_{01}(t) \right) < MTh$, a switching transition problem is detected.

One should note here that a 0->0 transition problem can simultaneously occur with a switch problem or a 1->1 transition problem. However, they cannot occur all 3 on the same node, since the sum of all mobilities is one. Similarly, a 1->1 transition problem can simultaneously occur with a switch problem.

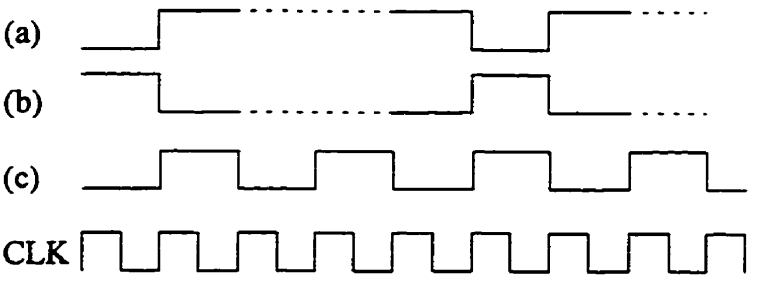


Figure 4.4 Examples where two control problems occur simultaneously

Figure 4.4 shows some cases where two control type problems can simultaneously occur. In (a), a 0->0 transition problem and a switching transition problem occur

simultaneously. In (b), a 1->1 transition problem and a switching transition problem occur simultaneously. By contrast, (c) shows a situation where 1->1 and 0->0 transition problems occur simultaneously. Note that cases (a) and (b) in Figure 4.4 are closely related to classical stuck-at-1 and stuck-at-0 control type problems.

In the following, we will discuss the effects of several test point structures on the mobility problems mentioned above.

AND and OR gates, excited on one of their inputs with an equiprobable pseudo-random source, are widely used to respectively eliminate 0-controllability and 1-controllability problems. In fact, if $c^l_{0b} < CTh$ (b stands for before insertion) for instance, after the insertion of an AND gate, the controllabilities of l will be (a stands for after insertion)

$$\begin{aligned} c^l_{0a} &= 0{,}5 + 0{,}5 \times c^l_{0b} \\ c^l_{1a} &= 0{,}5 - 0{,}5 \times c^l_{0b} \end{aligned} \tag{4.13}$$

as a consequence

$$\begin{aligned} 0{,}5 \le c^l_{0a} &\le 0{,}5 + 0{,}5 \times CTh \\ 0{,}5 - 0{,}5 \times CTh \le c^l_{1a} &\le 0{,}5 \end{aligned} \tag{4.14}$$

The new controllabilities lie between $0{,}5 - \frac{CTh}{2}$ and $0{,}5 + \frac{CTh}{2}$. However, after the insertion of an AND gate, the mobilities of l will be

$$
\begin{aligned}
m^{l}_{11a} &= 0{,}25 \times m^{l}_{11b} \\
m^{l}_{01a} &= 0{,}5 \times m^{l}_{01b} + 0{,}25 \times m^{l}_{11b} \\
m^{l}_{10a} &= 0{,}5 \times m^{l}_{10b} + 0{,}25 \times m^{l}_{11b} \\
m^{l}_{00a} &= 1 - \left(m^{l}_{11a} + m^{l}_{10a} + m^{l}_{01a} \right)
\end{aligned}
\tag{4.15}
$$

From equation (4.15), we note that AND gate insertion, which aims at eliminating a 0-controllability problem, may introduce a 1->1 transition problem if $m^{l}_{11b} < 4 \times MTh$. In this case, we find $m^{l}_{11a} < MTh$ after insertion. Similarly, for an OR gate, a 0->0 transition problem may be introduced if $m^{l}_{00b} < 4 \times MTh$. In conclusion, the classical AND/OR insertion may introduce 1->1 or 0->0 transition problems, which are not detected if we only rely on classical controllabilities for insertion.

By contrast, XOR gates are known to have better pseudo-random characteristics than AND/OR gates [126]. After insertion of an XOR gate, the mobilities of l will be

$$
\begin{aligned}
m^{l}_{11a} &= 0{,}25 \\
m^{l}_{01a} &= 0{,}25 \\
m^{l}_{10a} &= 0{,}25 \\
m^{l}_{00a} &= 0{,}25
\end{aligned}
\tag{4.16}
$$

XOR gates also have less impact on observability than AND/OR gates. In fact, the observability of each input of the XOR is equal to the observability of its output. Note, however, that this is not always exact. Figure 4.5 illustrates an example where the observability of B is altered as a consequence of inserting an XOR gate. Indeed, before the insertion, $A_1 = 0$ and $A_2 = 1$ is required to observe B. After insertion, $A_1 = 0$,

$A_2 = 1$ and $I = 0$ is required to observe B. Consequently, $o_a^B = \frac{o_b^B}{2}$, where o_b^B and o_a^B are observabilities of "B" before and after the insertion of the XOR gate respectively.

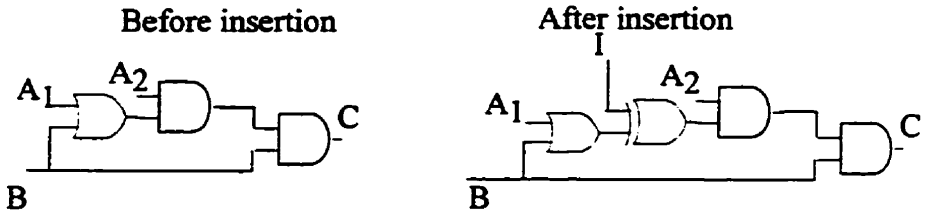


Figure 4.5 A test case where observability is altered as a consequence of inserting an XOR gate

In conclusion, XOR gates are good test point insertion structures. However, they are generally more complex than classical AND/OR gates (NAND/NOR gates). An implementation of XOR with only 6 transistors does exist, which is the same complexity as CMOS AND and OR gates, but it is associated with intrinsic testability and cascadability problems [115]. In the following, we propose a simple procedure for selecting among test point structures, which minimizes hardware overhead. In this procedure, we assume that XOR gates are more complex than AND/OR gates, and that they introduce larger delays. As a consequence, XOR gates are only used when required and the preference is always given to AND/OR gates.

Figure 4.6 sketches the procedure used to differentiate between the 3 types of mobility problems discussed above. The case where the 0->0 mobility is smaller than *MTh* and the 1->1 mobility is larger than 4 *MTh* is addressed with an AND gate, and that situation is called an AND-testable fault. Similarly, the case where the 1->1 mobility is smaller than *MTh* and the 0->0 mobility is larger than 4 *MTh* is addressed with an OR gate, and that situation is called an OR-testable fault. All remaining cases where mobilities are smaller or would become smaller than *MTh* are addressed with XOR gates, including the case where the 0->1 and 1->0 mobilities are inadequate. That later case where 0->1

and 1->0 mobilities are inadequate will be called switching fault of type 1. The case where both the 1->1 and 0->0 mobilities are too low, a type 2 switching fault, will be addressed by case 3. Indeed, the XOR gate can prevent the output from switching with a probability of 1/2 which is a solution to type 2 switching faults.

1. if($m^{l}_{00b} < MTh$ and $m^{l}_{11b} > 4 \times MTh$)
   Fault = AND Fault; insert an AND control point
2. else if($m^{l}_{11b} < MTh$ and $m^{l}_{00b} > 4 \times MTh$)
   Fault = OR Fault; insert an OR control point
3. else if($m^{l}_{01b} < MTh$ or $m^{l}_{00b} < MTh$ or $m^{l}_{11b} < MTh$)
   Fault = Switch Fault; insert a XOR control point

Figure 4.6 Procedure to determine the type of control problems

4.2.4 Joint Effect of Inserting Several Test Points

Up to now, we have considered the effect of each test point taken in isolation. Thus the joint effect of multiple test points was not considered. The joint effect is analyzed in the following. In Section 4.2.4.1, we show through an example that inserting multiple test points may result in better insertion methods. The well known split personality problem of classical test point structures is also considered in Section 4.2.4.1. The concept of multiple test points simultaneously inserted is suggested as a possible solution to this problem. In Section 4.2.4.2, we point out that multiple test point insertion may result in losses of fault coverage, if they are not adequately handled. These losses are primarily related to contradicting testability requirements between hard to test faults.

4.2.4.1 Detection with Multiple Test Points

Figure 4.7 illustrates a situation where the joint effect of several test points helps to eliminate hard to detect faults. This obviously results in more effective test points. In Figure 4.7, the controllabilities to 0 of A and B are very small. To test "C" (make mobilities of "C" > MTh), two test points are simultaneously needed: one at "A" and another one at "B". Indeed, the controllabilities to 0 of A and B are assumed to be so weak that a single test point in "A" does not cover point "C". Similarly, a single test point in "B" does not cover "C". However, "C" is detectable by the simultaneous insertion of test points at "A" and "B".

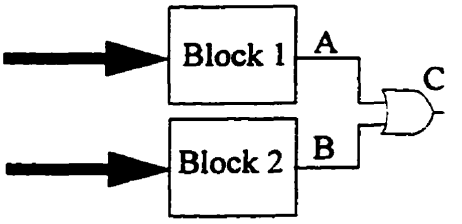


Figure 4.7 Example of multiple detection.

Finding the optimal solution for the "multiple test point" problem may be expensive. Indeed, the number of possible solutions to explore is $N!$ (N factorial) where N is the number of candidate test points. Moreover, for each solution, in order to minimize the number of test points, we may need to compute the incremental effect of each point for all possible orders, which is very expensive. In section 4.2.5, we will present a practical heuristic solution to this problem.

In some situations, solving a mobility problem that appears as a drastic decrease of mobilities in one stage may require multiple test points. Figure 4.8 shows an example of such situations. Let us assume that the mobilities of the inputs of the OR gate are acceptable and that those of the output A are poor. The insertion of a point in A will solve

the mobility problems of the output cone of A_2, but those of A_1 remain unsolved. To deal with this situation, we must insert test points at the inputs of the OR gate, and one or several points may be required. This problem, known as split personality, was already raised by different authors [15].

Note that inserting several test points on the inputs of gate A, to cover the test problems in the output cone of A, may not be as effective as inserting a single test point in A. As a consequence, we may need to insert a test point in "A", as well as other test points in some inputs of the gate feeding A. Further solutions to the split personality problems will be left for future development.

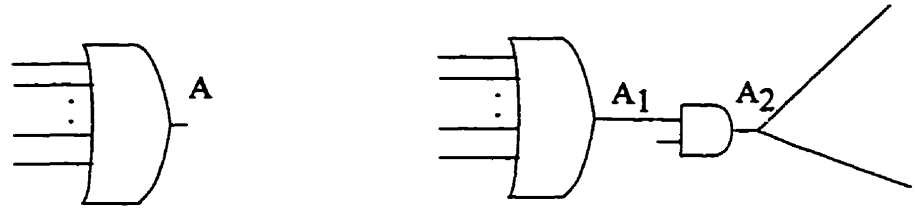


Figure 4.8 A fault may require simultaneous insertion of several points.

4.2.4.2 Test Point Compatibility

The simultaneous insertion of several points may result in more effective test point insertion methods, as mentioned above. However, in some cases, a test point could be associated with negative side effects on testability. An example of this situation is illustrated in Figure 4.9. Let us assume in this example that, in the original circuit, the 1-controllability of "B" is very small, and that due to other inputs, the controllability of A is acceptable. Since, there is a 1-controllability problem in "B" and no control problem in A, inserting a 1-control point in "B" may result in a test problem in A. Nodes B and A may have incompatible requirements. As a consequence, A and B may be difficult to test at the same time.

In general, this problem arises when several test points are simultaneously inserted. Their mutual effects may result in fault coverage losses due to incompatibilities between testability related requirements. A related problem has already been mentioned in several papers dealing with weighted random vectors [143]. In fact, some resistant faults may have different and contradicting requirements, and thus several different weights may be required on the same internal node to fully test a circuit.

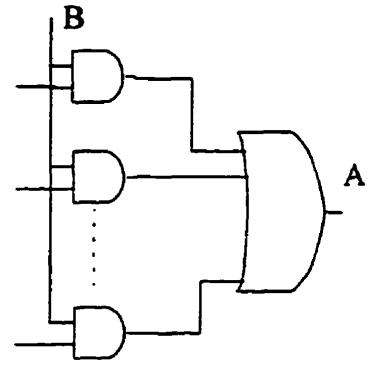


Figure 4.9 An example of a test point with negative side effect

In this paper, we propose to use different modes during testing. In each mode, only compatible test points are considered. The modes are activated in sequence one after another, such that no two modes are simultaneously active. Note that a test point will never be used in more than one mode, even though it may required to solve some split personality problems. The testing modes are proposed here to solve the conflicting requirements of multiple test points. Nevertheless, such multiple sequential test modes may have other advantages: they may be used to reduce the negative impacts on timing; they may also be used to increase pseudo-random sources sharing. In fact, the maximum number of pseudo-random sources needed is at most equal to the maximum number of test points required in one of the modes. The modes can also help to enhance I/Os sharing as proposed in [145] (for condensation). Power dissipation is another issue where different

modes can be helpful. Unfortunately, each additional mode is usually associated with additional overheads. A control state machine (counter, decoding logic) is needed to manage the test sequence (activating, deactivating the modes etc.). This may further impact the timing. Finally, test application time is increased, since the test should be restarted for each mode.

In conclusion, a good test point insertion method should produce a small set of test points grouped in a small number of test modes to cover all hard-to-detect faults.

4.2.5 Heuristics for Test Point Insertion

In the present and next sections, several heuristic rules for test point insertion are proposed. The aim of these heuristic rules is to find and insert a small number of test points to transform hard to test sequential circuits into easy to test ones, assuming they are tested with pseudo-random vectors. With these heuristics, controllability problems are targeted first. Observability problems are considered after having solved all controllability problems.

Heuristic M1

The first heuristic rule targets detection problems and builds the detection cones reflecting the effects of inserted test points on hard-to-control faults. As a first step, all the mobilities, the observabilities as well as the detectabilities (equation (4.12)) are computed. In a second step, all hard to detect faults are identified. A fault is declared hard to detect if its detectability computed with $N = 2^{15}$ is smaller than a threshold set arbitrarily to $1 - 10^{-6}$. The hard-to-control faults are then computed from the hard-to-detect faults list. A hard-to-detect fault is declared hard-to-control if one of its mobilities is below a predefined mobility detection threshold set arbitrarily to $ThldM = 0{,}005$. The hard-to-

control faults are included in a list called $HARD_1$. In the third step, for each hard to control fault X, we determine 3 lists: Det_X, $NDet_X$ and Pot_X. A fault $Y \in HARD_1$ is included in Det_X if its mobilities after insertion of a test point in X are larger than $ThldM = 0{,}005$. Det_X is called the detection cone of X. A fault $Y \in HARD_1$ is included in Pot_X if a positive enhancement has been detected in one or more of its mobilities, however, one (or more) of them is still lower than $ThldM$. Pot_X is called the potential detection cone of X. $NDet_X$ includes all the faults of the circuit which develop a controllability problem (one of its mobilities become inferior to $ThldM$) as a consequence of inserting a test point in X. $NDet_X$ represents the negative side effect of a point inserted in X. Figure 4.11 illustrates a typical partition of $HARD_1$ and the fault universe with respect to the 3 lists Det_X, $NDet_X$ and Pot_X for a given test point X.

Heuristic M1: Test problem detection

1. Compute all required probabilities of all nets in the original circuit.
2. Determine all hard-to-detect faults and the set of hard-to-control faults. The set of all hard-to-control faults is called $HARD_1$.
3. For each hard to control fault $X \in HARD_1$, compute the following sets:
   a. the set Det_X of faults of $HARD_1$ which become controllable as a consequence of inserting a test point in X.
   b. the set $NDet_X$ of faults in the circuit which become uncontrollable as a consequence of inserting a test point in X.
   c. the set Pot_X of faults $HARD_1$ which are potentially detected as a consequence of inserting a test point in X.
4. Deactivate the test point inserted in X and return back to 3 until all faults in $HARD_1$ are considered.

Figure 4.10 Phase 1: hard fault detection and the effect of their corresponding test point

Heuristic M2

The objective of the second heuristic is to find a small set $HARD_2$ of test points to cover all $HARD_1$ faults. This is the classical covering problem which is known to be NP complete. Several heuristic solutions have been proposed in the literature for this problem. In the following, we consider a simple solution based on the search of the test point associated with the largest Det_X. This solution produces acceptable results despite its simplicity.

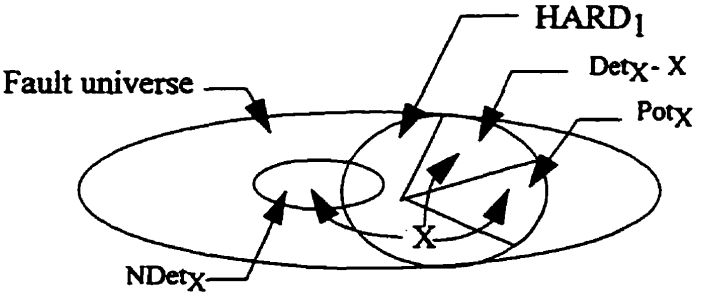


Figure 4.11 Illustration of the 3 sets: DetX, NDetX and PotX for a given test point X

One should note here that the set $HARD_2$ is a list of test points which is sufficient to cover all hard-to-control faults in $HARD_1$. However, since we only considered controllability problems, it does not necessarily guarantee the detection of all of them. Moreover, some test points in $HARD_2$ may be incompatible as mentioned in section 4.2.4.2.

Heuristic M2: Eliminate hard-to-control faults

1. find the test point associated with the largest Det_X and include "X" into $HARD_2$
2. remove all faults in Det_X from HARD1 and update all Det_Y of all remaining faults Y in $HARD_1$
3. return to 1, until $HARD_1$ is empty.

Figure 4.12 Phase 2. A heuristic to find a small subset of points which cover all $HARD_1$ faults.

Heuristic M3: Create different modes for incompatible test point sets and eliminate redundancies in $HARD_2$

1. build the potential detection tree as explained below,
2. rank the tree according to the potential detection relation as shown in Figure 4.15.
3. $i = 1$
4. put all faults of $HARD_2$ in mode M_i
5. starting from the lowest rank level and for all test point of M_i
   select a test point $X \in M_i$
   compute mobilities
   if no negative side effect,
      retain "X" in mode M_i
      eliminate all M_i test points fully detected (cone included) by the insertion of X
   else
      Deactivate X and assign it into another mode M_{i+1}
6. if M_{i+1} is not empty, increment i and return to 5, else exit.

Figure 4.13 Phase 3. A heuristic to eliminate redundancies between the test points in $HARD_2$ and to create different modes for incompatible test points.

Heuristic M3

The third phase (Figure 4.13) has two main objectives: the first one is to eliminate "redundant" test points[1] in $HARD_2$ (produced by heuristic M2) by considering the effect of simultaneous insertion of several test points. The second objective is to find a small number of modes such that each mode will only include compatible test points

In this heuristic, the test points in $HARD_2$ are first ranked according to their potential detection level as will be explained later when we present the concept of Potential Detection Tree (PDT). Then, considering the test points in increasing rank, test points from $HARD_2$ are inserted, one at a time, taking into account the effect of previously inserted test points, until no test point remains in $HARD_2$. Let us stress that in $HARD_2$ the joint effect of test points had not been taken into account. Therefore, in the third phase (Figure 4.13), when a test point is inserted, it will first be checked for incompatibilities. If a given test point is incompatible with the previously inserted ones, it will be assigned to another mode. A test point is said to be incompatible with previously inserted test points if one fault already covered becomes undetected as a consequence of inserting the test point in question. In this case, we also say that the fault is associated with a negative side effect. Secondly, all the remaining test points in $HARD_2$ are checked to determine if they have been covered as well as their respective cones by the previously inserted points. If this is the case, the test points (and their cones) which are already covered are eliminated from $HARD_2$. Figure 4.14 illustrate the effect of heuristic M3 on the set $HARD_2$. The set $HARD_3$ is the result after having applied heuristic M3. $HARD_3 \subseteq HARD_2$ is partitioned into N modes and of course $|HARD_3| \leq |HARD_2|$.

---

1. A test point is said redundant if it can be covered (including its cone) by multiple insertion of other test points.

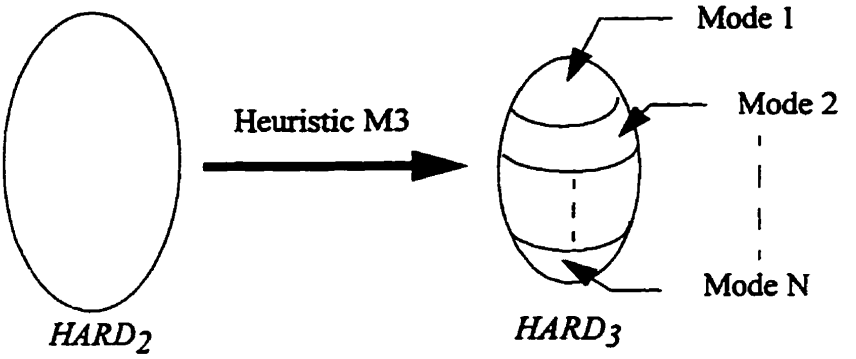


Figure 4.14 Effect of heuristic M3 on the set $HARD_2$

Before going further, let us define the concept of potential detection tree (PDT). The nodes of the tree are the test points of $HARD_2$. Each arrow between two nodes indicates that the test point at the end of the arrow is potentially or hardly detected by the test point at its source. For instance, in Figure 4.15, a test point in "A" potentially or hardly detects faults B and D. D has two in-nodes {A, G} and two out-nodes {E, I}. Note that there may remain some undetectable faults in $HARD_2$ due to the order in which faults are retained. Therefore, PDT takes into account hard to detect faults.

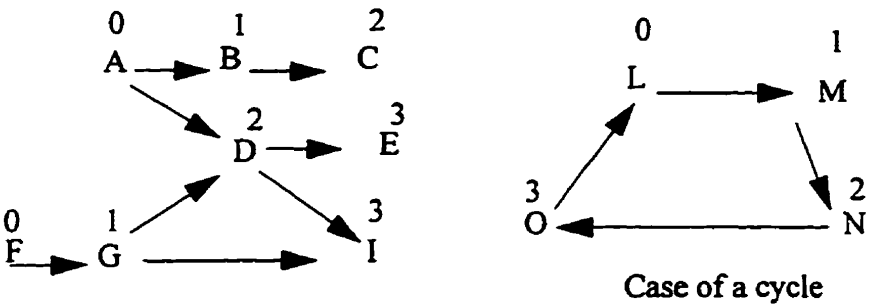


Figure 4.15 Examples of potential detection graphs

The tree is then ranked according to the potential detection relation as shown in Figure 4.15. This ranking is accomplished in a way similar to the method used to rank

logic circuits, where each node is assigned a rank equal to the maximum rank of its inputs plus one. Cycles are artificially cut before computing the ranks.

Heuristic M_3 should create "N" modes. Each mode only includes compatible test points. Note however, that in addition to these N modes, a mode M_0 where no point is inserted may be required. In fact, it may happen that some faults which were easy to test when no test point is inserted become untestable as a result of inserting some test points. In this case, we need a mode M_0 to cover these losses. Note here that heuristic M3 is complete in the sense that it considers the detection of all hard-to-control faults $HARD_1$, however, it does not guarantee detection, because there may be observability problems or faults that requires multiple test points (split personality problems).

Fault simulation also has an impact on the most appropriate order for using the various modes. Indeed, fault simulations can be very expensive, which led to the development of acceleration techniques in fault simulation tools. With these techniques, as the fault simulation advances, the simulated regions shrinks to the minimum required by the remaining untested faults. Therefore, if there is a mode M_0, it should not be invoked first, since it is expected to be less effective than mode M_1, not only in the reachable coverage, but also in the number of test vectors required to reach a given coverage.

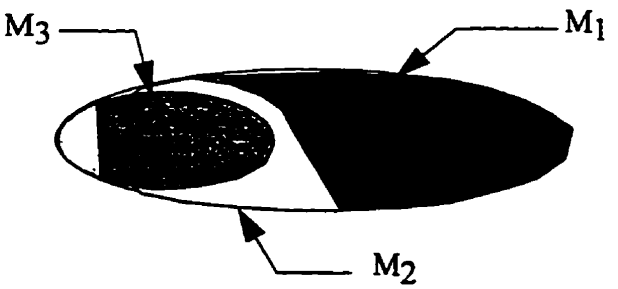


Figure 4.16 An example where 3 modes cover all the hard-to-detect faults

In the example shown in Figure 4.16, the order M_1, M_2, M_3 may be better than M_2, M_3, M_1. In fact, if almost all the faults are covered in modes M_1 and M_2, fault simulation in mode M_3 would be less expensive.

4.2.6 Observability Test Point Insertion

It is well known that observabilities are related to controllabilities, and by consequence to mobilities. An observability problem can be caused by a lack of mobility. Thus, observability insertion should be considered after having solved all mobility problems. Moreover, observability problems which may appear only in some modes but not in others should not be considered for observability insertion, unless all modes where there is sufficient mobilities for a fault have an observability problem. Faults linked to such problems will generally be detected in the modes where the joint mobility-observability problem does not appear.

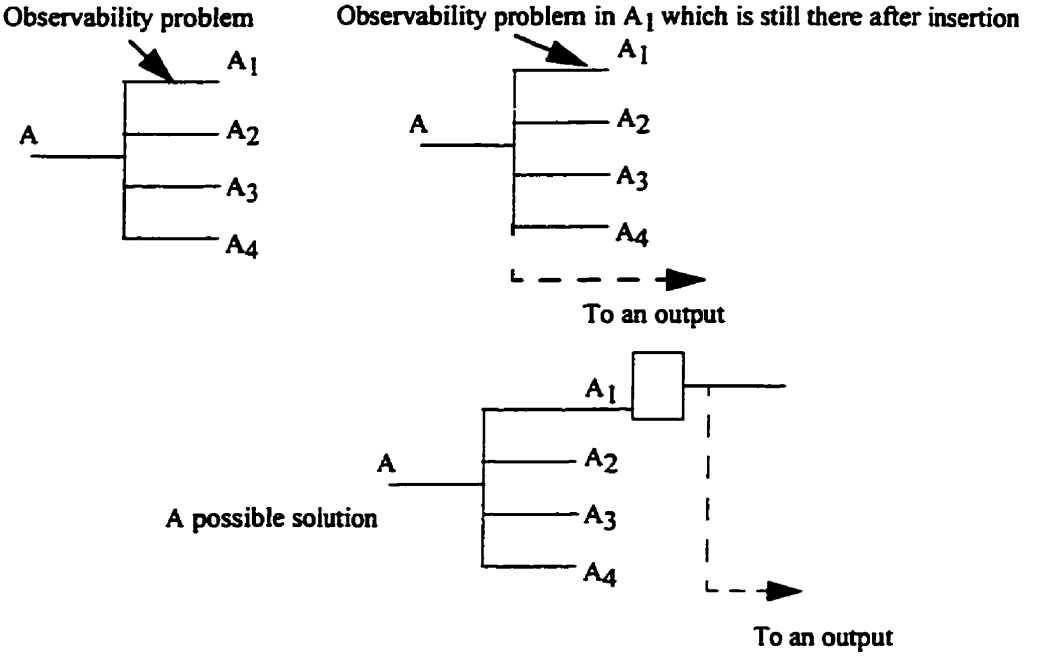


Figure 4.17 Impact of test points on fanout stems

Correcting the observability profile is simpler than correcting mobility profiles. Indeed, there is no conflicting requirement between observability points, and thus they do not introduce any negative side effect on the detectability as the mobility points can do. Therefore, there is no intrinsic need of different test modes for observability insertion. We may still wish to have different modes for other reasons, such as condensation or resource sharing, but not for pure testability reasons. However, observability insertion has some particularities. For instance, in Figure 4.17, we illustrate an example where the A_1 branch of fanout stem A has an observability problem. However, the root "A" may not have any observability problem, since it may be easily observed through the other fanout stems. Inserting an observation point in A will not solve the problem of A_1. It remains not observable after insertion.

A solution to this problem, which is similar to the split personality problem of mobilities (discussed above) is to push the insertion location to the output of the gate driven by the fanout stem (see Figure 4.17). However, it may happen that the observability of the output is good, but the mobilities of the other inputs of the same gate are only slightly superior to the mobility detection threshold. In this case, inserting an observability point on the output may not be a solution. In the present paper, we only considered the proposed solution (insertion at the output of the gate driven by the fanout stem). Further discussions on alternative solutions and their implementations are left for future research.

Heuristic O1: Observability problem detection

1. invoke mode M_0
2. compute all detectabilities
3. determine all hard to observe faults. The set of all hard to observe faults is called OBS_1.
4. for the remaining modes M_i
   Compute detectabilities

   Eliminate, from OBS_1, all observability problems which have been detected in mode M_i, except those corresponding to joint mobility-observability problems.

5. for each observability point in $X \in OBS_1$, individually compute the following sets:
   a. the set Det_X of faults of OBS_1 which become detectable as a consequence of inserting a single observability point in X.

   b. the set Pot_X of faults of OBS_1, which are potentially detectable as a consequence of inserting an observability point in X.

Figure 4.18 Heuristic for detecting observability problems

In the following, several heuristics similar to those used for control point insertion are proposed for observation point insertion. With the first heuristic, summarized in Figure 4.18, observability problems are located. In the first step of this heuristic, the first mode M_0 is activated, the detectabilities are computed (step 2) and the observability problems (step 3) are collected in OBS_1, from the hard-to-detect fault list. Note here that steps 1 and 2 are identical to steps 1 and 2 of heuristic M_1. They are repeated here just for clarity, however, they are not implemented in O_1. Then, in step 4, the remaining modes are activated one after another. During each mode, observability problems of OBS_1 which

disappear (in the considered mode) are eliminated, except those corresponding to a joint mobility-observability problem. At the end of step 4, OBS_1 will include the nodes with observability problems, which are not detected in any mode, and the joint mobility-observability problems. In step 5, the observation cones are individually built for all observation points in OBS_1.

Heuristic O2: Eliminate hard to observe faults

1. find the observability point associated with the largest Det_X (Det_X containing the largest number of faults) and include "X" into OBS_2
2. remove all faults in Det_X from OBS_1 and update all Det_Y of all remaining faults Y in OBS_1
3. return to 1, until OBS_1 is empty.

Figure 4.19 Heuristic for extracting a small subset of observability points that cover all observability problems in OBS_1

In heuristic O2, a small set of observability points are selected to cover all observability problems of OBS_1. Note that the cumulative effect of simultaneously inserting several observation points is not considered in heuristic O2. Therefore, redundancies may still exist in OBS_2. This heuristic is identical to heuristic M2 for control point insertion.

In the third heuristic, redundancies as a consequence of simultaneous insertion of several observability points are detected and eliminated. First, the potential observation tree is built. An observation point A is said to detect observation point B if inserting A results in enhancing the observability of B beyond an observability threshold $ThldO$, and it is said to potentially detect B if inserting A results in an enhancement of the

observability of B, but that such enhancement is not sufficient to raise observability beyond the threshold $ThldO$.

The potential observation tree in this case is similar to the potential detection tree built for mobility problems. The nodes of the tree represent the observation points of OBS_2, and each arrow between two nodes indicates that the observation point at the end of the arrow is potentially or hardly detected by the observation point at its source.

Heuristic O3: eliminate redundancies in OBS_2

1. build the potential observation tree,
2. rank the tree from outputs to inputs according to the potential observation relation
3. starting from the lowest rank and for all observation point in OBS_2
   select an observation point X

   compute observabilities

   eliminate all observation points detected by the insertion of X

Figure 4.20 Heuristic to eliminate redundancies in OBS_2

4.2.7 Experimental Results and Discussion

Based on the notions and heuristics presented, a prototype tool called SETIN[1] for automatic insertion of test points in sequential circuits was implemented in the C language. The SETIN tool reads a description of the circuit and compute the mobilities, observability and detectabilities of every fault into the circuit (see Figure 4.21). Then the program estimates the original fault coverage. According to this value, the program

1. SETIN from SEquential Test point INsertion

proceeds to the insertion of test points. First, all hard-to-control faults are analyzed to construct their three sectors defined above: Det_X, $NDet_X$ and Pot_X. Second, test point selection algorithms are invoked to determine a small set of candidates for insertion. These candidates, when individually inserted one after another, guarantee elimination (according to testability measures) of all controllability problems which can be solved by a test point at the source of the problem. Note that some controllability problems related to split personality phenomenon may remain on the fan-in of control test points. Then, the retained candidates are grouped into a small set of modes. At this stage, SETIN estimates the fault coverage to determine if further insertion (observability insertion) is required. Next, the program treats the hard-to-observe points as discussed in Section 4.2.6.

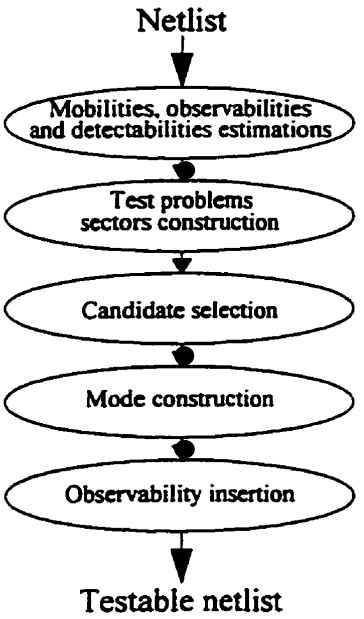


Figure 4.21 SETIN Block Diagram

Using SETIN, we performed a set of experiments on a large subset of the well known sequential ISCAS'89 benchmarks. Table 4.1 presents some statistics on these benchmarks. The "Seq" column gives fault coverages of the original circuits (without modification). These numbers were obtained with the fault simulator HOPE [74] using a

pseudo-random sequence of 2^{15} vectors. In some cases, 100% coverage was reached with less than 2^{15} vectors, in this case the number of vectors required to reach 100% is reported in parenthesis. Under the "comb" column, we report fault coverages of the same benchmarks after implementation of a full scan chain[1]. Since probabilistic testability measures are blind to redundant faults, and in order to avoid biasing the results, redundant faults should not be considered. Therefore, we need to determine a list of redundant faults. The number of redundant faults in each circuit is reported under the column "Redundant faults". These faults were obtained using the fault simulator Hope. All faults were stimulated with pseudo-random vectors, and those still undetected after applying 10^7 random test patterns were put in a list of faults which are declared redundant. For the sake of comparison, we also reported the number of redundant faults as produced by the ATPG Gentest [50]. Note that in some cases, such as s510, the method we used produces fewer redundancies than Gentest. This is primarily related to the fact that in order to fault simulate with Hope, all flip-flops are initialized to a known state. Consequently, some faults which are not detected by Gentest, where an unknown initial state is assumed, are easy to detect with fault simulation. For a more elaborate discussion on the problem of initializing sequential circuits with pseudo-random vectors refer to [129]

Table 4.1 Statistics on the ISCAS'89 benchmark circuits used here

Circuit name	DFF	Seq	Comb	Redundant Faults	
		FC(#V)	FC(#V)	Hope (10^7)	Gentest
s27	3	100 (28)	100(64)	0	0
s208	8	57.674	100 (6432)	74	78
s298	14	86.039	100 (384)	35	35
s344	15	99.123	100 (192)	3	9

1. Note here that fault coverages reported in Table 4.1 are not corrected for redundant faults.

Table 4.1 Statistics on the ISCAS'89 benchmark circuits used here

Circuit name	DFF	Seq	Comb	Redundant Faults	
		FC(#V)	FC(#V)	Hope (10^7)	Gentest
s349	15	98.571	99.429	5	10
s382	21	26.316	100 (320)	279	11
s386	6	69.010	100 (4608)	80	70
s420	16	30.698	96.512	226	251
s444	21	22.785	97.046	341	16
s510	6	100 (499)	100 (1088)	0	564
s526	21	19.820	99.820	416	41
s526n	21	19.892	100 (30944)	414	41
s641	19	87.580	98.929	58	63
s713	19	82.616	92.599	101	87
s820	5	50.471	100 (12480)	388	15
s832	5	49.655	98.391	406	15
s838	32	23.804	87.515	545	581
s953	29	99.073	99.907	10	991
s1196	18	98.390	99.597	3	3
s1423	74	58.878	98.944	309	8
s1488	6	75.572	100 (10496)	138	26
s1494	6	74.635	99.203	154	30
s5378	179	66.022	99.022	1411	118
s35932	1728	85.819	89.81	-	3999

In the first experiments, we ran SETIN using the following thresholds $ThldE = 1 - 10^{-7}$ (fault detection threshold), $ThldM = 0{,}005$ (Mobility threshold) and $ThldO = 0{,}0001$ (observability threshold). The results are reported in Table 4.3. "$HARD_1$" represents the total number of hard-to-control points declared by the testability measures. Under the column "$HARD_2$", we reported the number of test point candidates produced by heuristic 2 (see Figure 4.12). Under the column "$HARD_3$", we reported the results of heuristic 3 (see Figure 4.13). "Mode" and "Mob" respectively designate the number of modes and the final number of control points as produced by heuristic M3. Under the observability columns, we find "HO", the total number of hard-to-observe

faults as declared by SETIN, as well as the number of observation points required to solve the detected problems.

The following important observations can be made from Table 4.3. Heuristic M2, despite its simplicity, produces good results. Only few test points are generally required to cover all hard-to-control faults. For instance, in the case of s382, only 15 test points is required by heuristic M2 to address the 118 detected control problems. Heuristic M3 was able to further optimize the results in several cases with only few test modes. For example, in the case of s832, SETIN produces 33 test points with heuristic 2, and with heuristic 3, it was able to reduce this number to 19 in 2 modes.

Table 4.2 Test point insertion results

Circuit name	$HARD_1$	$HARD_2$	$HARD_3$		Observe		Hope Results			
			mode	Mob	HO	Obs	U1	FC1	U2	FC2
s208	69	14	2	12	19	3	0	100	0	100
s298	28	11	2	11	13	1	26 6	91.558 98.053*	0	100
s344	16	6	2	6	14	1	1	99.708	0	100
s349	17	7	2	7	9	1	3	99.143	0	100
s382	118	15	2	15	18	1	2	99.499	0	100
s386	50	19	2	18	36	2	20 17	94.792 95.57*	1	99.68
s420	161	32	2	28	75	9	14 9	96.744 97.91*	3	98.52
s444	130	11	2	10	7	1	15	96.835	0	100
s510	0	0	0	0	0	0	0	100	0	100
s526	157	36	4	29	91	8	49 31	91.171 94.41	1	99.80
s526n	159	37	4	37	93	6	69 66	87.568 88.10	1	99.80
s641	53	6	2	5	15	3	2	99.572	0	100
s713	56	3	2	2	0	0	38	93.460	0	100

Table 4.2 Test point insertion results

Circuit name	$HARD_1$	$HARD_2$	$HARD_3$		Observe		Hope Results			
			mode	Mob	HO	Obs	U1	FC1	U2	FC2
s820	69	43	2	43	127	2	69 43	91.882 94.94*	11	98.68
s832	64	33	2	19	6	1	64 47	92.644 94.60*	5	99.41
s838	343	38	2	31	32	32	90 66	89.498 92.30*	0	100
s953	37	5	3	4	1	1	7 6	99.351 99.44*	0	100
s1196	64	19	2	15	3	1	6 3	99.517 99.76*	0	100
s1238	69	10	2	22	5	1	43 33	96.827 97.56*	-	-
s1423	166	25	2	25	72	1	175 124	88.449 91.81*	-	-
s1488	20	16	2	16	5	1	47 43	96.837 97.11*	7	99.52
s1494	19	16	2	16	7	1	53 48	96.481 96.81*	10	99.32
s5378	1082	142	2	142	15	15	379 282	91.766 93.87*	0	100
s35932	1438	453	2	453	1112	1	4068 3993	89.594 89.78*	1904	94.57

* results obtained with mode M_0 activated

To validate the results, fault simulations with Hope using a pseudo-random sequence of 2^{15} test vectors were invoked and the results are reported in Table 4.3 under the column "Hope Results". The column "U1" gives the remaining number of undetected faults, and "FC1" the corresponding fault coverage. Then, the columns "U2" and "FC2" respectively designate the number of undetected faults and the corresponding fault coverage, after the results have been corrected by removing the redundant faults (redundant faults as declared by HOPE, see Table 4.1). A comparison between these results and those obtained with the

original circuits, as shown in Table 4.1, demonstrates the effectiveness of the proposed heuristics to alleviate testing problems in sequential circuits. In fact, almost complete fault coverage have been obtained in most cases, when the results are corrected by removing redundant faults as reported in Table 4.1. Moreover, we can easily notice that several redundant faults have been accidently detected after insertion. For instance, in the case of s713, the fault coverage increased from 82.61% to 93.46% by inserting only 2 test points in 2 modes. The number of undetected faults after insertion is 38 and Gentest reported 87 redundant faults. FC2 is 100% in this case.

One should note that the mode M_0 which corresponds to the original circuit (with all test points deactivated) is systematically considered in these experiments. This mode was shown to be worthless in some cases. For instance, in the case of s444, 96.83% of fault coverage was obtained by activating all the test points in mode M_1, M_0 did not contribute to the fault coverage. However, in the case of s298, M_0 was able to reduce the number of undetected fault from 26 to 6.

By removing the redundant faults, we notice that the fault coverage "FC2" is lower than "FC1" in some cases. For instance, in the case of s420, "FC1" is 99.30% and FC2" is 98.52%. This is simply because the statistics were biased by the fact that several redundant faults were made detectable by test point insertion. These faults are taken into account in "FC1" but not in "FC2".

The results reported in Table 4.3 depend on the selected mobility and observability thresholds. Experiments on the impact of varying these thresholds were carried out, and Table 4.3 reports the values which produce the best coverage results when varying the thresholds between 0.1 and 0.00001. In Table 4.3, "U" and "FC" respectively designate the minimum number of undetected faults, and the corresponding fault coverage as

produced by Hope with a test sequence of 2^{15} vectors. These numbers have been corrected by removing redundant faults from the original fault lists (redundant faults as declared by HOPE, see Table 4.1). ThldM and ThldO are the mobility and observability thresholds under which these results were obtained.

Table 4.3 Best test point insertion results

Circuit name	mode	Mob	Obs	FC	ThldM	ThldO	Muradali [90]	
							C/O	FC(100k)
s298	2	8	1	100	.001	.0001	-	-
s344	2	4	1	100	.001	.0001	-	-
s349	2	4	1	100	.001	.0001	-	-
s382	2	13	1	100	.001	.0001	-	-
s386	2	14	1	99.68	.001	.0001	-	-
s420	2	19	4	100	.0005	.00001	7/0	100
s444	2	9	0	100	.0005	.00001	10/0	97.4
s526n	2	9	8	99.8	.001	.001	7/5	97.0
s641	2	4	3	99.76	.001	.0001	1/0	100
s820	2	13	2	99.78	.0001	.0001	3/2	100
s832	2	12	1	99.41	.0001	.0001	5/12	100
s838		14	18	100	.00005	.0005	13/0	100
s1423		32	4	98.23	.01	.0001	25/17	97.8
s5378		83	15	100	.001	.0001	42/26	97.7

From Table 4.3, we notice that better results may be obtained by varying the thresholds. For instance, in the case of s420, a fault coverage of 100% instead of 97.91% was obtained. Surprisingly, the number of test points required for testing in this case decreased from 28 to 19 for mobility points, and from 9 to 4 for observability points. In general, we noticed that test point insertion methods are fairly sensitive to the selected thresholds. A good set of threshold should accurately differentiate testing problems. Severe thresholds may escape some interesting test sites, resulting in bad insertion decisions, and relaxed thresholds may produce unnecessary test points, which may result

in losses of fault coverage. The dynamic of test quality as a function of the number of test points seems to follow the curve depicted in Figure 4.22.

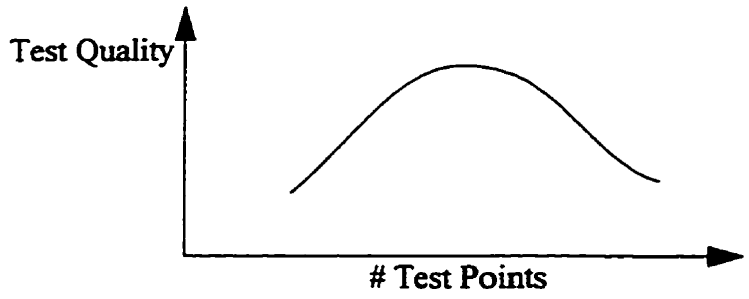


Figure 4.22 Dynamic of test quality as a function of the number of test points

Furthermore, we also compare in Table 4.3 our results with those obtained by Muradali and Rajski [90]. "C/O" and "FC" respectively represent the number of control / observation points and the corresponding fault coverage they obtained with 100K pseudo-random test vectors. Table 4.3 shows that, in several cases, SETIN is able to achieve better results. For instance, in the case of s526n, with SETIN, we obtain 99.8% fault coverage comparatively to 97% with the Muradali and Rajski method. Let us recall that our test sequence length is only 2^{15} vectors as compared to 100K vectors in [90]. However, it turns out that, in several cases, SETIN produces more test points than Muradali's and Rajski's method. In fact, their method is based on logic simulation, which is known to be more accurate in assessing testing problems. Nevertheless, logic simulation-based methods are more expensive. Indeed, in order to get an acceptable precision, large number of vectors are usually required, particularly for large circuits. Unfortunately, since Muradali and Rajski did not report time complexity we cannot compare that important aspect of the two methods.

4.2.8 Conclusion

In this paper, a novel multiphase test point insertion method for sequential circuits has been proposed. This method is based on the mobility measure, which was shown to better quantify testing problems in sequential circuits. The mobility measure is based on transition probabilities instead of steady-line probabilities. The phases proposed here are sequentially activated one after another. In each phase, a subset of selected control points are inserted. When constructing the phases, the joint effect of several test points considered. As a consequence, conflictual requirements between several faults are automatically detected and the corresponding test points are assigned to different modes. Experimental results show that it is possible to achieve good results with the proposed method, with only a small number of test points distributed between few phases.

CHAPITRE 5

Conclusion

La présente thèse constitue à notre avis un avancement de notre compréhension de la testabilité séquentielle pseudo-aléatoire des circuits VLSI. Nous avons étudié et analysé dans cette thèse les problèmes de testabilité séquentielle, de plus, nous avons proposé un ensemble de solutions pratiques qui s'inscrit dans le cadre d'un outil informatique d'insertion de points de test pour le test pseudo-aléatoire des circuits séquentiels.

Au chapitre 2, nous avons étudié le problème de l'initialisation des circuits séquentiels dans un contexte de test pseudo-aléatoire. Nous avons proposé un modèle basé sur les chaînes de Markov pour modéliser le phénomène de l'initialisation. Avec ce modèle, nous avons démontré que l'initialisation avec des vecteurs pseudo-aléatoires est possible dans plusieurs cas. Le problème de l'initialisation a été abordé par d'autres auteurs. L'originalité de ce chapitre se situe à notre avis dans la démonstration que l'initialisation en fonction de la longueur de la séquence d'initialisation pseudo-aléatoire est une fonction croissante. Autrement dit, il est exclu de retourner à un état inconnu une fois qu'une machine séquentielle est initialisée. Un circuit facile à initialiser oublie l'état de départ après un certain nombre de vecteurs et ce nombre est généralement limité. Dans le chapitre 2, nous avons également vu des cas de circuits séquentiels impossibles ou difficiles à initialiser par des vecteurs pseudo-aléatoires. Nous avons montré comment les détecter par les mesures de testabilité sCOP et nous avons proposé des techniques de reset partiel afin de les transformer en circuits faciles à initialiser.

Nous estimons ici qu'une étude sur les problèmes des pannes qui empêchent l'initialisation est une extension logique des travaux présentés au chapitre 2. Les pannes qui empêchent l'initialisation furent étudiées dans un contexte de test algorithmique, mais jamais dans un contexte de test pseudo-aléatoire. Une autre extension du chapitre 2 pourrait être de poursuivre l'étude d'autres heuristiques plus efficace de *reset* partiel et d'insertion de points d'initialisation combinatoires.

Au chapitre 3, nous avons abordé le problème de la détection des pannes dans les circuits séquentiels. Ainsi, nous avons montré les limites des mesures sCOP relativement au problème de la détection des pannes. Nous avons donné des exemples pratiques de circuits séquentiels simples tel que le compteur binaire où les mesures sCOP échouent dans la quantification précise des problèmes de test des différents noeuds. Ceci est relié au fait que les mesures sCOP sont aveugles aux corrélations temporelles, très importantes dans les circuits séquentiels. Pour remédier à ce problème, nous avons proposé une nouvelle mesure de testabilité appelée la mobilité. Les mesures de mobilité sont basées sur le calcul des probabilités de transition. Leur complexité de calcul reste linéaire en fonction du nombre de portes d'un circuit avec un facteur de linéarité un peu plus complexe que celui des mesures sCOP. Avec les mesures de mobilité, nous sommes en mesure de couvrir plusieurs pannes et problèmes de test des circuits séquentiels. Les résultats expérimentaux ont montré qu'elles sont systématiquement plus précises que les mesures sCOP.

Avec les mesures de mobilité, certaines corrélations temporelles entre deux cycles consécutifs sont détectées. Cependant, nous savons que d'autres corrélations d'ordre supérieur ne sont pas détectables avec les mesures de mobilité. Une corrélation sur 3 cycles ou plus sur le même noeud, ainsi qu'une corrélation spatiale sur 2 noeuds ou plus sont des pathologies non détectables par les mesures de mobilité. Étudier ces pathologies et propo-

ser un ensemble de métriques faciles à calculer pour les détecter constitue un developpement futur d'un grand intérêt pour la communauté du test.

Nous avons également montré au chapitre 3 qu'avec les mesures de mobilité, nous pouvons faire une estimation approximative de la consommation de puissance moyenne. En plus de son importance dans la phase de conception des circuits VLSI modernes, cette estimation est de plus en plus nécessaire dans un contexte de test pseudo-aléatoire. En effet, l'activité anormalement élevée des noeuds internes lors d'un test pseudo-aléatoire mérite une attention particulière si nous ne voulons pas brûler notre puce avant de terminer le test. Nous avons montré dans ce chapitre que les erreurs sur l'estimation de la puissance introduites par les mesures de mobilité ne sont que de l'ordre de 10% sur une longue liste de circuits d'essai. De plus, l'erreur relative tend à diminuer avec l'augmentation de la complexité des circuits.

Une extension logique de cette section du chapitre 3 consistera à définir un ensemble de règles pour faire une gestion adéquate de la consommation de la puissance durant un test pseudo-aléatoire. Autrement dit, établir des règles de partitionnement et une cédule (séquence de test) en tenant compte des exigences de la dissipation de puissance.

Au chapitre 4, nous avons étudié le problème de l'insertion des points de test dans les circuits séquentiels. Nous avons utilisé les mesures de mobilité décrites au chapitre 3, pour prédire et localiser les problèmes de testabilité ainsi que pour suivre l'évolution de la testabilité lors de l'insertion. Nous avons également étudié des notions qui à notre connaissance n'ont jamais été traitées dans la littérature reliée à l'insertion des points de test. Nous avons ainsi étudié l'effet de l'insertion conjointe sur les performances des méthodes d'insertion de points de test. L'insertion conjointe semble avoir des effets bénéfiques ainsi que des effets indésirables. Des solutions heuristiques basées sur une méthodologie multi-

phase furent proposées. Dans chaque phase, seulement les points compatibles sont invoqués.

Nous estimons que ce chapitre est un pas dans le domaine de l'insertion de points de test dans les circuits séquentiels. nous pensons que d'autres heuristiques plus raffinées peuvent être développées en tenant compte de la qualité du test, des performances du circuits, de la circuiterie ajoutée, du nombre de ports d'entrée/sortie additionnels ainsi que de la rapidité d'exécution de ces heuristiques (par des techniques d'élagage par exemple). Dans ce chapitre, nous avons proposé une méthodologie multiphase où chaque phase a ses propres points de test activés et désactivés en bloc. Nous estimons qu'une méthodologie insertion multiphase où les phases sont entrelacées peut être plus efficace dans certaines situations. Par exemple, une panne qui nécessite deux points de test dans deux phases différentes peut être efficacement supportée par une méthodologie multiphase où les phases sont entrelacées.

Références

[1] ABRAMOVICI, M., BREUER, M. A. et FRIEDMAN, A. D., (1990), Digital Systems Testing and Testable Design, Revised Printing, IEEE PRESS, ISBN. 0-7803-1062-4..

[2] ABRAMOVICI, M., KULIKOWSKI, J., et ROY, R. K., (1991), The Best Flip-Flops to Scan, Proceedings of IEEE International Test Conference, 166-173,.

[3] ABRAMOVICI M., et PARIKH, P. S., (1992), Warning: 100% Fault Coverage May Be Misleading, Proc. of International Test Conference

[4] ABRAMOVICI, M., PARIKH, P. S., MATHEW, B., et SAAB, D. G., (1993), On Selecting Flip-Flops for Partial Reset, Proc. of IEEE International Test Conference, 1008-1012

[5] AGRAWAL, V.D., et al., (1989),An Economical Scan Design for Sequential Logic Test Generation, Proceedings of the 19th International Symposium on Fault Tolerant Computing, 118-125.

[6] .AGARWAL, V. D.., et CERNY, E.,(1981), Store and generate Built-In Self-Testing, Proc. FTCS 11, 35-40.

[7] AGRAWAL, V.D.., et. CHAKRABORTY, T.J., (1995), High-performance Circuit Testing with Slow Speed Testers, Proceedings of International Test Conference, 302-310.

[8] AGRAWAL, V.D., CHENG, K-T., JOHNSON, D. D et LIN, T. (1988), Designing Circuits with Partial Scan, IEEE Design & Test of Computers, 8-15.

[9] AGRAWAL, V. D., CHENG, K-T., JOHNSON, D. D., et LIN, T., (1987), A Complete Solution to the Partial Scan Problem, Proceedings of IEEE International Test Conference, 44-51.

[10] AGRAWAL, V., et MERCER, M.R., (1982) Testability Measures-What Do They Tell Us? Proc. International Test Conference, 391-396.

[11] AGRAWAL, V. D., SETH, S. C., et AGRAWAL, P., (1981), LSI Product Quality and Fault Coverage, ACM/IEEE Design Automation Conference, 196-203.

[12] AMAZONAS, J. R., et STRUM, M., (1989), Pseudoexhaustive Test Techniques: A New Algorithm to Partition Combinational Networks, Proceedings of the 1st European Test Conference, 392-397.

[13] ANDO, H., (1980), Testing VLSI with Random Access Scan, Proc. of COMP-CON S'80, 50-52.

[14] AYARI, B., BEN HAMIDA, N., et Kaminska, B., Random and Deterministic Test Vector Generation for Sequential Circuits, submitted for publication in IEEE Transactions on CAD.

[15] BARDELP, H., McANNEY, W. H., et SAVIR, J., (1987), Built-In Test For VLSI Pseudorandom Techniques, a Wiley Inter-Science Publication, John Wiley & Sons ISBN. 0-471-62463-2.

[16] BEN HAMIDA, N., et KAMINSKA, B., (1993), Initiability: A Measure of Sequential Testability, Proc. of International Symposium on Circuits and Systems, Chicago.

[17] BEN HAMIDA, N., et KAMINSKA, B., (1991), A Uniform Testability Measure Representation for Sequential and Combinational Circuits, IEEE International Symposium on Circuits and Systems, 1984-1987.

[18] BEN HAMIDA, N., KAMINSKA, B., et SAVARIA, Y., (1994), Pseudo-random Vector Compaction for Sequential Testability, Proc. of International Symposium on Circuits and Systems, 63-66.

[19] BEN BENNETTS, R. G., (1994), Progress in Design for Test: A personal View, IEEE Design and Test of Computers, 53-59.

[20] BENNETTS, R. G., MAUNDER, C. M., et ROBINSON, G. D., (1981), CAME-LOT: A Computer-Aided Measure for Logic Testability, IEE Proc., Vol. 128, Part E, No. 5, 177-189.

[21] BOUBEZARI, S., et KAMINSKA, B., (1996), A New Reconfigurable Test Vector Generator for Built-In Self-Test Applications, Journal of Electronic Testing: Theory and Applications, Vol. 8, 153-164.

[22] BRACKEN, R. C., SALATINO, M. M., ADKINS, C. L., et KRAEMER, B. P. (1991), Multi-Chip-Modules: A Comparative Study: Phase 1. System Design and Substrate Selection, University/Government/Industry Microelectronics Symposium, Melbourne Florida.

[23] BREWER, J.. E., (1994), A Single-Chip Digital Signal Processing Subsystem, International Conference on Wafer Scale Integration, 265-272.

[24] BREWER, M. A., (1978), New Concepts in Automated Testing of Digital Circuits, Proc. EEC Sym. on CAD of Digital Electronic Circuits and Systems, Brussels, 69-92.

[25] BRIERS, A. J. , et TOTTON, K.E., (1986), Random Pattern Testability by Fast Fault Simulation, Proceedings of the IEEE International Test Conference, 274-281.

[26] BRGLEZ, F., BRYAN, D., et KOZMINSKI, K., (1989). Combinational Profiles of Sequential Benchmark Circuits, International Symposium on Circuits and Systems.

[27] BRGLEZ, F., POWNAL, P., et HUM, R., (1985). Accelerated ATPG and Fault Grading via Testability Analysis," in Proc. of IEEE International Symposium on Circuits and Systems, pp. 695-698, 1985.

[28] BRGLEZ, F., POWNAL, P., et HUM, R., P., (1984), Applications of Testability Analysis: from ATPG to Critical Delay Path Tracing, Proc. IEEE International Test Conference, 705-712.

[29] CANADIAN MICROELECTRONICS CORPORATION, (1990), The CMOS4S Standard Cell Library," 26-32.

[30] CANADIAN MICROELECTRONICS CORPORATION, (1989), The CMOS3S Standard Cell Library," 26-32.

[31] CHENG, T-T., et AGRAWAL, V., (1992), Initializability Consideration in Sequential Machine Synthesis, IEEE Transactions on Computers, Vol. 41, pp. 374-379.

[32] CHENG, T-T., et AGRAWAL, V., (1990), A Partial Scan Method for Sequential Circuits with Feedback, IEEE Transactions, Vol. 39, No. 4, 544-548,

[33] CHENG, T-T., et AGRAWAL, V., (1989), An Economical Scan Design for Sequential Logic Test Generation, Proceedings of the IEEE Fault Tolerant Computing Symposium, 28-35.

[34] CHENG, T-T., et AGRAWAL, V., (1988), Designing Circuits with Partial Scan, IEEE Design and Test of Computers, 8-15.

[35] CHENG, T-T., et BREUER, A., (1985),Automatic Design for Testability via Testability Measures, IEEE Transactions on Computer-Aided Design, Vol. CAD-4, 3-11.

[36] CHENG, D. I., CHENG, K-T., WANG, D., et MAREK-SADONSKA, M., (1996) A New Hybrid Methodology for power Estimation, 33rd ACM/IEEE Design Automation Conference.

[37] CHENG, K.-T., et LIN, C.-J., (1995) Timing-Driven Test Insertion for Full-Scan and Partial-Scan BIST, Proceedings of Internationa Test Conference, 506-514.

[38] CHIKERMANE, V., et PATEL, J.-H., (1991), A Fault Oriented Partial Scan Design Approach, Proc. of International Conference on Computer-Aided Design, 332-335.

[39] CHIKERMANE, V., et PATEL, J.-H., (1990), An Optimization Based Approach to the Partial Scan Design problem, Proceedings of IEEE International Test Conference, 377-386.

[40] CHIN, C. K., et McCLUSKEY, E. J., (1985), Test Length for Pseudo-Random Testing, Proceedings of the IEEE International Test Conference, 94-99.

[41] DARLAY, F., SOUFI, M., SAVARIA, Y., et KAMINSKA, B., (1992), "Pseudo-random Vectors Can Achieve Deterministic Initialization," Canadian Conf. on VLSI.

[42] DASGUPTA, S., GOEL, P., WALTER, R.C., et WILLIAMS, T., (1982), "A Variation of LSSD and its Implication on Design and Test Pattern Generation in VLSI," International Test Conference, 63-66.

[43] EICHELBERGER, E.B., et WILLIAMS, T.W., (1977), "A Logic Design Structure for LSI Testability," Proc. of the 14th Design Automation Conference, 206-212.

[44] ERCOLANI, S., FAVALLI, M., DAMIANI, M., OLIVO, P., et RICCO, B., (1992), "Testability Measures in Pseudorandom Testing," IEEE Transactions on CAD, Vol. CAD-11, 794-800.

[45] FOX, J. R. , (1977), "Test Point Condensation in the Diagnosis of Digital Circuits," Proceedings IEE, Vol. 124, No. 2.

[46] FUJIWARA, H. et YAMAMOTO, A., (1992) "Parity-scan Design to Reduce the Cost of Test Application," Proceedings of International Test Conference, 283-292,

[47] FUNATSU, S., WAKATSUKI, N., et YAMAD, A., (1978), "Designing Digital Circuits with Easily Testable Considerations," Proc. of International Test Conference, 98-102.

[48] GAEDE, R., MERCER, M. R. et UNDERWOOD, B., (1996), "Calculation of Greatest Lower Bounds Obtainable by the Cutting Algorithm," Proc. International Test Conference, 498-505.

[49] GAGE, R., (1993), "Structured CBIST in ASICs," Proceedings of International Test Conference, 332-338.

[50] AT&T GENTEST ATG ISCAS'89, (1989), BENCHMARK RESULTS,

[50] AT&T GENTEST ATG ISCAS'89, (1989), BENCHMARK RESULTS,

[51] GHOSH, A., DEVADAS, S., KEUTZER, K., et WHITE, J., (1992), Estimation of Average Power Switching Activity in Combinational and Sequential Circuits, 29th ACM/IEEE Design Automation Conference, 253-259.

[52] GOEL, P., (1980), Test Cost Analysis and Projections, Proceedings of 17th Design Automation Conference, 77-84.

[53] GOLDSTEIN, L. M., et Thigen, E. L., (1980), SCOAP: Sandia Controllability/ Observability Analysis Program, Proc. 17th Design Automation Conference, 190-196.

[54] GOLDSTEIN, L. H., (1979), Controllability/Observability Analysis of Digital Circuits, IEEE Transactions on Circuits and Systems, Vol. CAS-26, No. 9,685-693.

[55] GRASSON, J., (1979), TMEAS, A Testability Measurement Program, 16th Design Automation Conference, 156-161.

[56] GUNDLACH, H. H., et MULLER-GLASSER, K-D., (1990), On Automatic Test Point Insertion in Sequential Circuits, Proceedings of the IEEE International Test Conference, 1072-1079.

[57] GUPTA, R., GUPTA, R., et BREUER, M. A., (1990), The BALLAST Methodology for structured Partial Scan Design, IEEE Transactions on Computers, Vol 39. No. 4, 538-54.

[58] HAYES, J. P., (1974), On Modifying Logic Networks to Improve their Diagnosability, IEEE Transactions on Computers, Vol. C-23, No. 1.

[59] HAYES, J. P., et FRIEDMAN, A. D., (1974), Test Point Placement to Simplify Fault Detection, IEEE Transactions on Computers, Vol. C-23, 727-735.

[60] HAYES, J.P., et McCLUSKEY, E. J., (1980), Testability Considerations in Microprocessor-Based Designs, Computer.

[61] HUISMAN, L. M., (1988), The Reliability of Approximate Testability Measures, IEEE Design and Test of Computers, 57-67.

[62] INTELLIGEN, (1989), HHB: Intelligen documentation package, Mahwah, New Jersey.

[63] IYENGAR, V. S., et BRAND, D., (1989), Synthesis of Pseudo-Random Pattern Testable Design, Proceedings of the IEEE International Test Conference, 501-508.

[64] JAIN, S., et AGRAWAL, V., (1984), STAFAN: An Alternative to Fault Simulation, Proc. of Design Automation Conference, 18-23.

[65] JONE, W.-B., et PAPACHRISTOU, C.A., (1988), On Partioning for Pseudo-exhaustive Testing of VLSI Circuits, Proceedings of the IEEE International Test Conference on Circuits and Systems, 1843-1846.

[66] KAMINSKA, B., SAVARIA, Y., YOUSSEF, M., et KOUDIL, M., (1991), Test Point Insertion for Pseudo-Random Testing, Proceedings of the IEEE International Test Conference on Circuits and Systems.

[67] KARSNIESKI, A., (1991), Can Redundancy Enhance Testability, Proceedings of IEEE International Test Conference, 483-491.

[68] KARSNIESKI, A., et PILARSKI, S., (1989), Circular Self-Test path: A Low-Cost BIST Technique for VLSI Circuits, IEEE Transactions on Computers-Aided Design, 46-55.

[69] KIM, S., BANERJEE, P., et PATIL, S., (1991), A Layout Driven Design For Testability Technique for MOS VLSI Circuits, Proceedings of IEEE International Test Conference, 157-165.

[70] KIM, S., et KIME, C., R., (1993), Partial Scan Using Reverse Direction Empirical Testability, Proceedings of IEEE International Test Conference, 498-506.

[71] KONEMANN, B., et al., (1979), Built-In Logic Block Observation Techniques, International Test Conference 37-41

[72] KONEMANN, B., MUCHA, J., et ZWIEHOFF, G., (1980), Built-In Test for Complex Digital Integrated Circuits, IEEE Journal of Solid State Circuits, 315-318.

[73] KRISHNAMURTY, B., (1987), A Dynamic Programming Approach for the Test Point Insertion Problem, Proc. of the 24th Design Automation Conference, 695-705.

[74] LEE, H.K., et HA, D.S., (1992), HOPE: A Fast Parallel Fault Simulator for Sequential Circuits, Proceedings of ACM/IEEE Design Automation Conference, 336-340.

[75] LIN, C-J., ZORIAN, Y., et BHAWMIK, S., (1993), PSBIST: A Partial Scan Based Built-In Self Test Scheme, Proc. of IEEE International Test Conference, 507-516

[76] LISANKE, R., (1987), Finite State Machines Benchmark Set, Preliminary Benchmark Collection.

[77] LISANKE, R., BRGLEZ, F., DEGEUS, A.J., et GREGORY, D., (1987), Testability-Driven Random Test Pattern Generation, IEEE Transactions on Computer-Aided Design, Vol. CAD-6, 1082-1087.

[78] LSI LOGIC, (1993), LCA300k Gate Array 5V SeriesProduct Databook, LSI LOGIC .

[79] LSI LOGIC, (1986) Databook and Design Manual, HCMOS Macrocells, Macrofunctionc, LSI LOGIC.

[80] MARCULSEAU, R., MARCULSEAU D., et PEDRAM, M., (1995), Efficient Power Estimation for Highly Correlated Input Stream, Proceedings of Design Automation Conference, 628-634.

[81] MATHEW, B., et SAAB, D. G.,(1993), Partial Reset: An Inexpensive Design for Testability Approach, Proc. of the European Design Automation Conference with the European Event in ASICs Design, 151-155.

[82] MATHEW, B., et SAAB, D. G.,(1993), Augmented Partial Reset, International Conference on Computer-Aided Design, 716-719.

[83] MAXWELL, P.C., AITKEN, R.C., JOHNSON, V., et CHIANG, L., (1991), The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better Than 90%? Proceedings of International Test Conference, 358-364 .

[84] MAZUMDER, P., et PATEL, J.H., (1987), An Efficient Built-In Self Testing for Random Access Memory, Proceedings of the IEEE international Test Conference, 1072-1077.

[85] McCLUSKEY, E. J., (1984), Verification Testing - A pseudo-Exhaustive Test Technique, IEEE Transactions on Computers,.

[86] MEHTRA, H., BORAH, M., OWENS, R. M., et IRWIN, M. J., (1995),Accurate Estimation of Combinational Activity, Proceedings of Design Automation Conference, 618-623.

[87] MENTOR GRAPHICS, (1992), CMOSN Cell Notebook," Vol. 1, Version 8.2, Mentor Graphics .

[88] MERCER, M. R., et UNDERWOOD, B., (1984), Correlating Testability with Fault Detection Proc. International Test Conference, 697-704.

[89] MONTEIRO, J., DEVADAS, S., et LIN, B., (1994), A Methodology for Efficient Estimation of Switching Activity in Sequential Circuits, 31st Design Automation Conference, 12-14.

[90] MURADALI, F., AGARWAL, V., et NADEAU-DOSTIE, B., (1990), A New Procedure for Weighted Random Built-In Self-Test, Proc. of International Test Conference, 660-669.

[91] MURADALI, F., NISHIDA, T., et SHIMUZU, T., (1995), A Structure and Technique for Pseudorandom based Testing of Sequential Circuits, Journal of Electronic Testing: Theory & Applications, 107-115.

[93] MUROGA, S., (1982), VLSI System Design: When and How to Design Very-Large-Scale-Integrated Circuits, Willey-Interscience, New York (N. Y.).

[94] NADEAU-DOSTIE, B., et al., (1990), Scan Design Software for ASICs, Proc. of Canadian Conference on VLSI, 3-8, Vancouver.

[95] NAJM, F. N., (1995), Feedback, Correlation and Delay Concerns in the Power Estimation of VLSI Circuits, Proceedings of Design Automation Conference, 612-617.

[96] NAJM, F. N., (1993), Transition Density: A New Measure of Activity in Digital Circuits, IEEE Transactions on CAD, Vol. 12, No. 2, 310-325.

[97] NAJM, F. N., GOEL, S. et HAJJ, N., (1995), Power Estimation in Sequential Circuits, Proceedings of Design Automation Conference, 635-640.

[98] OHLETZ, M., WILLIAMS, J., et MUCHA, J.P., (1991), Overhead in Scan and Self-Testing Design, Proc. of IEEE International Conference on Computer-Aided Design, 376-380.

[99] PAPOULIS, A., (1991), Probability Random Variables and Stochastic Processes, Third Edition McGraw Hill.

[100] PARKER, K.P., and McCLUSKEY, E. J. (1975), Probabilistic Treatment of General Combinational Circuits, IEEE Transactions on Computers, 668-670 .

[101] Discussion personnelle avec Jacob Savir, Workshop de Kiwaha 1993.

[102] PIXELEY, C., JEONG, S-W., et HACHTEL, G. D., (1992) Exact Calculation of Synchronization Sequences Based on Binary Decision Diagrams, Proc. of 29th ACM/IEEE Design Automation Conference.

[103] PIXELEY, C., et BEIHL, G., (1991), Calculating Resetability and Reset Sequences, Proc. of IEEE International Conference on Computer-Aided Design, 376-380.

[103] PIXELEY, C., et BEIHL, G., (1991), Calculating Resetability and Reset Sequences, Proc. of IEEE International Conference on Computer-Aided Design, 376-380.

[104] RATIU, I., SANGIOVANNI-VINCENTELLI, M.A., et PETERSON, D.O., (1982) VICTOR: A Fast VLSI Testability Analysis Program, Proc. International Test Conference, 397-401.

[105] RHO, J-K., SOMENZI, F., et PIXELEY, C., (1993), Minimum Length Synchronizing Sequences of Finite State Machine, Proc. of 30th ACM/IEEE DESIGN Automation Conference.

[106] ROSS, S., (1989), Itroduction to Probability Models, 4th Edition, Academic Press, Boston, Toronto.

[107] ROTH, J. P., (1966), Diagnosis of Automata Failures: A Calculus and a Method, IBM Journal of Research and Development, 278-291.

[108] RUDNICK, E. M., CHIKERMANE, V., et PATEL, J. H., (1994),An Observability Enhancement Approach for Improved Testability and At-Speed Test, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 8, pp. 1051-1056.

[109] RUTHMAN, R. A., (1992), Fault Detection Test Generation for Sequential Logic Heuristic Tree Search, IEEE Computer Repository Paper, No. R-72-187.

[110] SAAB, D. S., SAAB, Y. G., et ABRAHAM, J. A., (1992), CRIS: A Test Cultivation Program for Sequential VLSI Circuits, Proc. of the Int. Conf. on Computer-Aided Design, 216-219 .

[111] SARMA, S., (1991), Built-In Self-Test Considerations in High-Performance, General-Purpose Processor, Proceedings of IEEE International Test Conference, 21-27.

[112] SAVARIA, Y., (1988), Conception et Vérification des Circuits VLSI, Chapître 6, Editions de l'École Polytechnique de Montréal, ISBN 2-553-00207-, 276.

[114] SAVARIA, Y., LAGUE B., et KAMINSKA, B., (1989), A Pragmatic Approach to the Design of Self-Testing Circuits, Proceedings of the IEEE International Test Conference, 745-754.

[115] SAVARIA, Y., YOUSSEF, M., KAMINSKA, B., et KOUDIL, M., (1991), Automatic Test Point Insertion for Pseudo-random Testing, Proc. of European Test Conference, 253-262.

[116] SAVIR, J.,"Good Controllability and Observability Do Not Guarantee Good Testability," IEEE Transactions on Computers, Vol. C-32, No 12, pp. 1198-2000, December 1983.

[117] SAVIR, J., DITLOW, G., et BARDELL, P., (1983), Random Pattern Testability, Proc. Fault Tolerant Computing Symposium, 80-89.

[118] SCHOTTEN, C., et MEYR, H., (1995), Test Point Insertion for Area Efficient BIST, Proceedings of International Test Conference, 515-523.

[119] SEISS, B. H., TROUBOST, P. M., et SCHULZ, M. H. (1991), Test Point Insertion for Scan-Based BIST, Proceedings of European Test Conference, 253-262.

[120] SETH, S., PAN, L., et AGRAWAL, V., (1985), PREDICT: Probabilistic Estimation of Digital Circuit Testability, Proc. of International Fault Tolerant Computing Symposium, 220-225.

[121] SHEDLETSKY, J. J., et McCLUSKEY, E. J., (1976), The Error Latency of a Fault in Sequential Digital Circuits, IEEE Transactions on Computers, Vol. C-25, No. 6, 655-659.

[122] SHEDLETSKY, J. J., et McCLUSKEY, E. J., (1975), The Error Latency of a Fault in Combinational Digital Circuits, 5th Annual Fault-Tolerant Computing Symposium, 210-214, Paris, France.

[123] SOUFI, M., ROCHON, S., SAVARIA, Y., et KAMINSKA, B., (1996). Design and Performance of CMOS TSPC Cells for High Speed Pseudo Random Testing, Proceedings of VLSI Test Symposium, 368-373.

[124] SOUFI, M., SAVARIA, Y., et KAMINSKA, Y., (1996), "An Iterative Calculation Method for Efficient Estimation of the Activity in Large Sequential Circuits," submitted for publication to the IEEE Transactions on VLSI.

[125] SOUFI, M., SAVARIA, Y., et KAMINSKA, B., (1995), On Using Partial Reset for Pseudo-Random Testing, Proc. of International Symposium on Circuits and Systems, 949-952.

[126] SOUFI, M., SAVARIA, Y., et KAMINSKA, B., (1995), On the Design of At-Speed Testable VLSI Circuits, Proceedings of VLSI Test Symposium, 290-95

[127] SOUFI, M., SAVARIA, Y., KAMINSKA, B., et DARLAY, F., (1996), Mobility Measure for Pseudo-random Testing of Sequential Circuits, submitted for publication to IEE.

[128] SOUFI, M., SAVARIA, Y., KAMINSKA, B., et DARLAY, F., (1994), Producing Reliable Initialization and Test for Sequential Circuits, technical report No. EPM-RT-94/23, Ecole Polytechnique of Montreal.

[129] SOUFI, M., SAVARIA, Y., KAMINSKA, B., et DARLAY, F., (1993), A Pseudo-Random Testing Approach for both Sequential and Combinational Circuits, Proc. of the IEEE International Conference on VLSI, Dhahran 16-19.

[130] SYNOPSYS Inc. (1996), Power Compiler Redefines Synthesis, RSVP Spring Edition.

[131] STEPHENSON, J. E., et GRASON, J., (1976), A Testability Measure for Register Transfer Level Digital Circuits, Proc. International Symposium on Fault-Tolerant Computing, 101-107.

[132] TAMARAPALLI, N et RAJSKI, J., (1996), Constructive Multi-Phase Test Point Insertion for Scan-Based BIST, Proc. of IEEE International Test Conference, 649-657.

[133] TRISCHLER, E., (1983), Testability Analysis and incomplete Scan Path, Proc. of International Conference on Computer-Aided Design, 38-39.

[134] TRISCHLER, E., (1980), Incomplete Scan Path with an Automatic Test Generation Methodology, Digest of Papers 1980 Test Conference, 153-162.

[135] TOUBA, N. A, et McCLUSKEY, E. J., (1995), Test Point Insertion Based on Path Tracing, Proc. International Test Conference.

[136] TOUBA, N. A., et McCLUSKEY, E. J., (1994), Automated Logic Synthesis of Random Pattern Testable Circuits, Proc. of IEEE International Test Conference, 174-183.

[137] TSUI, C-Y., MONTEIRO, J., PEDRAM, M., DEVADAS, S., DESPAIN, A.M., et LIN, B., (1995), Power Estimation Methods for Sequential Logic Circuits, Transactions on VLSI Systems, Vol. 3, No 3, 404-416.

[138] TSUI, C-Y., PEDRAM, M., et DESPAIN, M., (1994), Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs, 31st Design Automation Conference, 18-23.

[139] WAGNER, K. D., et al., (1987), Pseudo-Random Testing, IEEE Transactions on Computers, Vol. c-36, No. 3, 332-343.

[140] WAICUKAUSKI, J.A., (1985), A Statistical Calculation of Fault Detection Probabilities by Fault Simulation, Proc. of IEEE International Test Conference, 779-784.

[141] WAICUKAUSKI, J. A., et LINDBLOOM, E., (1988), Fault Detection Effectiveness of Weighted Random Patterns, Proc. of International Test Conference, 245-255.

[142] WANG, L. T., et McCLUSKEY, E. J., (1986), Condensed linear Feedback Shift register (LFSR) Testing: A Pseudo-Exhaustive Test Technique, IEEE Transactions on Computers, C-35, 367-370.

[143] WUNDERLICH, H-J., (1988), Multiple Distributions for Biased Random Test Patterns, Proc. of International Test Conference, 236-244.

[144] WUNDERLICH, H-J, (1985), PROTEST: A Tool for Probabilistic Testability Analysis, Proceedings of ACM/IEEE Design Automation Conference, 204-211.

[145] YOUSSEF, M., (1991), Insertion de points de test et implantation des chaines de balayage partielles dans les circuits séquentiels, Mémoire de Maîtrise, département de génie électrique, Ecole Polytechnique de Montréal.

[146] YOUSSEF, M., SAVARIA, Y., et KAMINSKA, B., (1993), A Methodology for Efficiently Inserting and Condensing Test Points, IEE Proceedings-E, Vol. 140, No. 3, pp. 154-160.

Annexes

Annexe 1. Calcul de la circuiterie ajoutée utilisé dans la section 2.2 pour estimer les coût d'un *reset* complet

Supposons que la circuiterie ajoutée par le mécanisme de *reset* est A_{Reset} portes par bascule. Supposons également que le nombre de portes totale par bascule avec mecanisme de *reset* est

$$Reset = Norm + A_{Reset} \tag{A.1.1}$$

ou $Norm$ est le nombre de portes dans une bascule normale (sans mecanisme de *reset*).

Le nombre total de portes P_{Norm} dans un circuit contenant N bascules sans chaîne de *reset* est

$$P_{Norm} = C + Norm \times N \tag{A.1.2}$$

où C est le nombre total de portes dans la partie combinatoire du circuit.

Le nombre total de portes P_{Reset} dans le même circuit avec un *reset* complet est

$$P_{Reset} = C + Reset \times N \tag{A.1.3}$$

Par conséquent, la circuiterie ajoutée due à un reset complet est

$$O_{Reset} = \frac{A_{Reset}}{K + Reset} \tag{A.1.4}$$

avec $K = \frac{C}{N}$.

Exercice

Si $O_{scan} = 30\%$ et $Reset = 90\% \times Scan$ ou $Scan$ est le nombre total de portes dans une bascule avec mecanisme de balayage, caculer la circuiterie ajoutée $O_{Reset} = \frac{A_{Reset}}{K + Reset}$ due à un reset complet ?

Du moment où $Reset < Scan$, nous pouvons ecrire

$$O_{Reset} = \frac{A_{Reset}}{K + Reset} > \frac{A_{Reset}}{K + Scan} \tag{A.1.5}$$

à partir de l'équation (A.1.1), l'équation (A.1.5) peut être reformulé comme suit

$$O_{Reset} > \frac{Reset - Norm}{K + Scan} \tag{A.1.6}$$

en remplacant $Reset = 90\% \times Scan$, nous retrouvons

$$O_{Reset} > \frac{0{,}9 \times Scan - Norm}{K + Scan} \tag{A.1.7}$$

de même, en remplacant $Scan = Norm + A_{Scan}$, nous retrouvons après factorisation,

$$O_{Reset} > \frac{0{,}9 \times A_{Scan}}{K + Scan} - \frac{0{,}1 \times Norm}{K + Scan} \tag{A.1.8}$$

le deuxième terme de cette inégalité est au maximum égale à 0.1. En effet, nous savons déjà que $Norm < Scan + K$, par conséquent le deuxième terme de l'inégalité (A.1.8) peut s'écrire come suit

$$\frac{0{,}1 \times Norm}{K + Scan} < \frac{0{,}1 \times (Scan + K)}{K + Scan} = 0{,}1 \tag{A.1.9}$$

finalement, O_{Reset} est au minimum supérieur à 0.17 comme montré ci-dessous

$$O_{Reset} > 0{,}9 \times 0{,}30 - 0{,}1 = 0{,}17 \tag{A.1.10}$$

Annexe 2. sCOP Computation Formulas

OR gate:

- $C_1(S,t)=1-(1-C_1(A,t))*(1-C_1(B,t))$
- $C_0(S,t)=C_0(A,t)*C_0(B,t)$
- $C_x(S,t)=1-(C_0(S,t)+C_1(S,t))$.

NAND gate:

- $C_0(S,t)=C_1(A,t)*C_1(B,t)$.
- $C_1(S,t)=1-(1-C_0(A,t))*(1-C_0(B,t)$.
- $C_x(S,t)=1-(C_0(S,t)+C_1(S,t))$.

NOR gate:

- $C_0(S,t)=1-(1-C_1(A,t))*(1-C_1(B,t))$
- $C_1(S,t)=C_0(A,t)*C_0(B,t)$
- $C_x(S,t)=1-(C_0(S,t)+C_1(S,t))$.

XOR gate:

- $C_0(S,t)=C_0(A,t)*C_0(B,t)+C_1(A,t)*C_1(B,t)$
- $C_1(S,t)=C_0(A,t)*C_1(B,t)+C_1(A,t)*C_0(B,t)$
- $C_x(S,t)=1-(C_0(S,t)+C_1(S,t))$.

XNOR gate:

- $C_1(S,t)=C_0(A,t)*C_0(B,t)+C_1(A,t)*C_1(B,t)$
- $C_0(S,t)=C_0(A,t)*C_1(B,t)+C_1(A,t)*C_0(B,t)$
- $C_x(S,t)=1-(C_0(S,t)+C_1(S,t))$.
-
-
-

Flip-flop:

- $C_1(Q,t)=C_1(D,t-1)$
- $C_0(Q,t)=C_0(D,t-1)$
- $C_x(Q,t)=1-(C_0(Q,t)+C_1(Q,t))$.

Annexe 3. Sketch of the proof that P(t) is non-decreasing

Assuming, N is the number of flip-flops in the circuit C, in the general case, the matrix "M" is given by

$$M = \begin{bmatrix} R & 0 \\ T & L \end{bmatrix} = \left[\begin{array}{ccc|ccc} m_{11} & m_{21}\cdots & m_{1(2^N)} & 0 & 0 & 0 \\ m_{21} & m_{22}\cdots & \cdots & \cdots & & \cdots \\ \cdots & \cdots & & \cdots & \cdots & \cdots \\ m_{(2^N)1} & \cdots & m_{(2^N)(2^N)} & 0 & 0\cdots & 0 \\ \hline m_{(2^N+1)1} & m_{(2^N+1)2}\cdots & m_{(2^N+1)(2^N)} & m_{(2^N+1)(2^N+1)} & \cdots & m_{(2^N+1)(3^N)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \\ m_{(3^N)1} & \cdots & m_{(3^N)(2^N)} & m_{(3^N)(2^N+1)} & \cdots & m_{(3^N)(3^N)} \end{array}\right]$$

where R is the transition matrix in the recurrent states subset, L the transition matrix in the transient subset, and T the transition matrix between the two. The recurrent states are numbered from 1 to 2^N and the transient states are numbered from $(2^N + 1)$ to (3^N).

Similarly, if we suppose that $\Pi(t) = (\alpha_1, \alpha_2, \alpha_3 \ldots, \alpha_1)$, then:

$$\Pi(t+1) = \Pi(t) \times M = \left(\sum_{i=1}^{3^N} \alpha_i m_{i1}, \ldots\ldots \sum_{i=1}^{3^N} \alpha_i m_{i(2^N)}, \sum_{i=1}^{3^N} \alpha_i m_{i(2^N+1)}, \ldots\ldots, \sum_{i=1}^{3^N} \alpha_i m_{i(3^N)} \right)$$

Therefore,

$$P(t+1) = \sum_{i=1}^{3^N} \alpha_1 m_{i1} + \ldots\ldots + \sum_{i=1}^{3^N} \alpha_i m_{i(2^N)}$$

$$= \alpha_1 \left(\sum_{j=1}^{2^N} m_{1j} \right) + \ldots\ldots + \alpha_{2^N} \left(\sum_{j=1}^{2^N} m_{(2^N)j} \right) + rest$$

Where $rest = \alpha_{(2^N+1)}\left(\sum_{j=1}^{3^N} m_{(2^N+1)_j}\right) + \ldots\ldots + \alpha_{(3^N)}\left(\sum_{j=1}^{3^N} m_{(3^N)_j}\right)$

Note that the sums between the parentheses in $P(t+1)$ are the sum of lines in the submatrix R, therefore $\sum_{j=1}^{2^N} m_{1j} = \sum_{j=1}^{2^N} m_{2j} = \ldots = \sum_{j=1}^{2^N} m_{(2^N)_j} = 1$

Hence, $P(t+1) = \alpha_1 + \alpha_1 + \ldots + \alpha_1 + rest$ where $rest \geq 0$

$= (P(t) + rest)$

$\geq P(t)$

Thus $P(t)$ is a non-decreasing function of "t". The weaker conclusion follows from the fact that even when the circuit is fully initializable, sequential circuit latency may hold the rest to zero for some time duration, when all transient states with a non-zero transition probability themselves have a zero probability. Practical examples of this behavior can be built with shift registers.

Annexe 4. Proof of theorem 1 on page 73

Letting "u" and "d" be two random variables denoting the number of cycles the Markov chain remains in states 1 and 0 respectively, without making any transition to the other state. As a consequence, we can write

$$U = E[u]$$
$$D = E[d]$$

The random variable $u + d$ may be considered as the number of clock cycles required to make the first round-trip from a state to itself through the second state. Then, since there is a single instance of each transition polarity every $u + d$, it follows that, on average, the rate "T" at which transitions occur is

$$T = \frac{1}{E[u+d]}$$

However,

$$E[u+d] = E[u] + E[d] = U + D$$

As a consequence,

$$T = \frac{1}{U+D}$$

Moreover, the rate at which we enter state 0 from state 1 " t_{10}" is given by

$$t_{10} = c_1 \times p_{10}$$

This rate is thus the rate at which a transition (1->0) occurs. As a consequence,

$$\frac{1}{U+D} = c_1 \times p_{10} \qquad \text{(A.4.1)}$$

To obtain a second equation, consider the percentage of time the process is in state 1, which is of course equal to c_1. Since the chain is in state 1 on average U out of every U+D time units, it follows that

$$\text{Proportion of time in state 1} = \frac{U}{U+D} = c_1 \qquad \text{(A.4.2)}$$

Hence, from equations (A.4.1) and (A.4.2), we can obtain

$$U = \frac{c_1}{c_1 \times p_{10}} = \frac{1}{p_{10}}$$

similarly, we can compute

$$D = \frac{1}{p_{01}}$$

Q.E.D

Annexe 5. Proof of Lemma 1 on page 76

Proof:

The rate "t_{ij}" at which the process enters a state $j \in sub_1$ from state $i \in sub_0$ is

$$t_{ij} = \pi_i \times p_{ij}$$

and thus the rate "$t_{sub_0 j}$" at which we enter state j from any state $i \in sub_0$ is

$$t_{sub_0 j} = \sum_{i \in sub_0} \pi_i P_{ij}$$

hence the rate $t_{sub_0 sub_1} = t_{01}$ at which we enter the subset sub_1 from sub_0 is

$$t_{01} = \sum_{(j \in sub_1)} \sum_{(i \in sub_0)} \pi_i P_{ij}$$

This rate is the rate at which a transition (0 -> 1) occurs on the flip-flop output "l". As a consequence,

$$m_{01}(l) = t_{01} = \sum_{(j \in sub_1)} \sum_{(i \in sub_0)} \pi_i P_{ij} \qquad \text{(A.5.1)}$$

In general, expression (A.5.1) may be used for any node "l" where "i" and "j" are respectively all the states which put the node "l" in the value 0 or 1. Expressions similar to the previous one generally apply to any node for all transitions of interest.

Since sequential circuits are, by assumption, modeled using ergodic Markov chain processes, the limiting probability vector $\Pi = \lim_{t \to \infty} \Pi(t)$ always exists and therefore $m_{01}(l)$ always exists.

Q.E.D.

Annexe 6. Mobility Formulas

OR gate

$$
\begin{aligned}
m_{00}(S) &= m_{00}(A) \times m_{00}(B) \\
m_{01}(S) &= m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{00}(B) + m_{00}(A) \times m_{01}(B) \\
m_{10}(S) &= m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{00}(B) + m_{00}(A) \times m_{10}(B) \\
m_{11}(S) &= 1 - (m_{00}(S) + m_{10}(S) + m_{10}(S))
\end{aligned}
\qquad \text{(A.6.1)}
$$

NAND gate

$$
\begin{aligned}
m_{00}(S) &= m_{11}(A) \times m_{11}(B) \\
m_{01}(S) &= m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B) \\
m_{10}(S) &= m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{11}(B) + m_{11}(A) \times m_{01}(B) \\
m_{11}(S) &= 1 - (m_{00}(S) + m_{10}(S) + m_{10}(S))
\end{aligned}
\qquad \text{(A.6.2)}
$$

NOR gate

$$
\begin{aligned}
m_{11}(S) &= m_{00}(A) \times m_{00}(B) \\
m_{01}(S) &= m_{10}(A) \times m_{10}(B) + m_{10}(A) \times m_{00}(B) + m_{00}(A) \times m_{10}(B) \\
m_{10}(S) &= m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{00}(B) + m_{00}(A) \times m_{01}(B) \\
m_{00}(S) &= 1 - (m_{11}(S) + m_{10}(S) + m_{10}(S))
\end{aligned}
\qquad \text{(A.6.3)}
$$

XOR

$$
\begin{aligned}
m_{00}(S) &= m_{00}(A) \times m_{00}(B) + m_{01}(A) \times m_{01}(B) + m_{10}(A) \times m_{10}(B) + m_{11}(A) \times m_{11}(B) \\
m_{01}(S) &= m_{00}(A) \times m_{01}(B) + m_{01}(A) \times m_{00}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B) \\
m_{10}(S) &= m_{00}(A) \times m_{10}(B) + m_{01}(A) \times m_{11}(B) + m_{10}(A) \times m_{00}(B) + m_{11}(A) \times m_{01}(B) \\
m_{11}(S) &= 1 - (m_{00}(S) + m_{10}(S) + m_{10}(S))
\end{aligned}
\qquad \text{(A.6.4)}
$$

Annexe 7. Exact Iterative Method for the Computation of the Steady Line Transition Probabilities

Ergodic Markov chain processes are often used for modeling the state transition diagrams of synchronous sequential circuits [75][129]. Figure App.7.1, for instance, shows an example of a Markov chain process corresponding to the MCNC benchmark, "dk15" [76]. The fractions represent the probabilities that the primary input values cause specific transitions.

The exact steady state probabilities can be computed using the well-known Chapman-Kolmogorov equations. The ergodicity assumption guarantees the existence of a unique solution for the steady state probabilities.

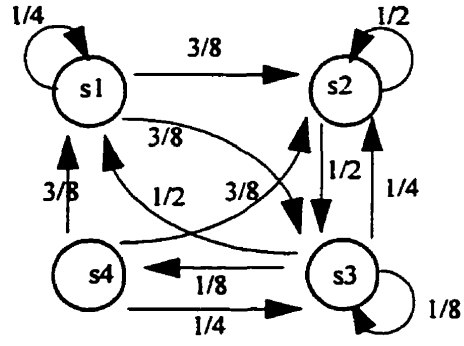


Figure A.7.1 Example of a sequential circuit (dk15) with its corresponding Markov chain diagram

Let $\pi_i(t)$ be the probability that the Markov chain is in state "i" after "t" cycles (i.e. after the application of "t" random input patterns to the circuit). Consequently the state probability vector of a sequential circuit with "n" flip-flops (2^n states) is defined as.

$$\Pi(t) = \left(\pi_1(t), \pi_2(t), \ldots\ldots \pi_{2^n}(t)\right) \qquad \text{(A.7.2)}$$

Then, $\Pi(t)$ can be obtained using the initial probability vector $\Pi(0)$, and the transition probability matrix "P", as follows:

$$\Pi(t) = \Pi(0) \times P^t \tag{A.7.3}$$

or, alternately, using the following iterative equation:

$$\Pi(t+1) = \Pi(t) \times P, \qquad \text{for } t \geq 0 \tag{A.7.4}$$

where P is defined as the matrix of individual transition probabilities in one cycle between states "i" and "j":

$$(P = [p_{ij}]) \ , \qquad i,j = 1, 2....., 2^n \quad .$$

The ergodicity assumption mentioned previously guarantees the existence of the limiting probability vector defined as

$$\Pi = \lim_{t->\infty} \Pi(t) \tag{A.7.5}$$

In the example in Figure App.7.1, for instance, the limiting probabilities are

$$\pi_{s_1} \approx 0{,}24, \quad \pi_{s_2} \approx 0{,}38, \quad \pi_{s_3} \approx 0{,}34, \quad \pi_{s_4} \approx 0{,}04$$

The exact steady line probabilities of the FF output (h_i) may be computed directly from their corresponding steady state probabilities as follows:

$$C_k^{h_i}(t) = \sum_{s_j \in sub_k} \pi_{s_j}(t) \tag{A.7.6}$$

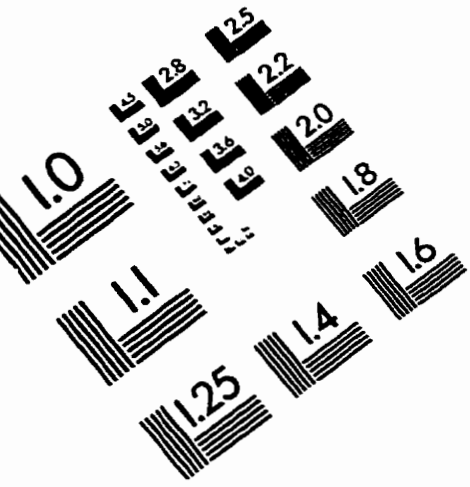
where $k \in \{0, 1\}$ and sub_k is the half state space, where the FF output h_i is at k.

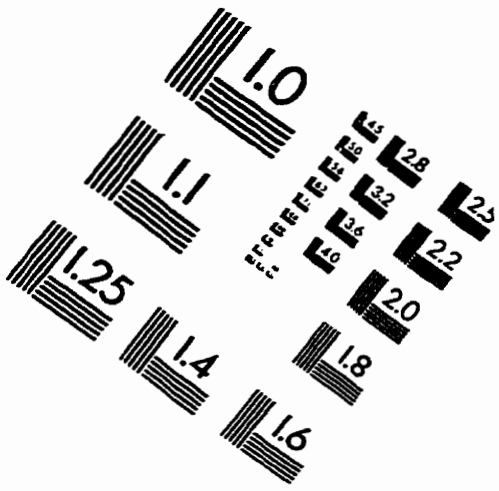
Annexe 8. Comparision between Typical formulas for computing controllabilities and mobilities

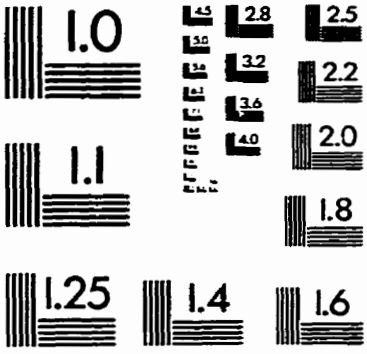
Table A.8.1 Typical formulas for computing controllabilities and mobilities

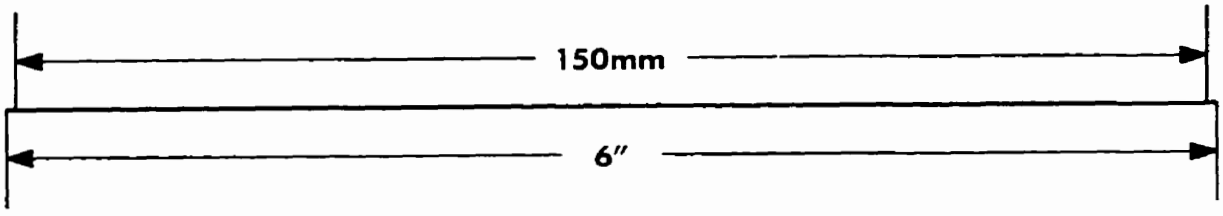
Controllabilities	Mobilities
AND $c_1(S) = c_1(A) \times c_1(B)$ $c_0(S) = 1 - c_1(S)$	$m_{11}(S) = m_{11}(A) \times m_{11}(B)$ $m_{01}(S) = m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{11}(B) + m_{11}(A) \times m_{01}(B)$ $m_{00}(S) = 1 - \left(m_{01}(S) + m_{10}(S) + m_{11}(S)\right)$
OR $c_0(S) = c_0(A) \times c_0(B)$ $c_1(S) = 1 - c_0(S)$	$m_{00}(S) = m_{00}(A) \times m_{00}(B)$ $m_{01}(S) = m_{01}(A) \times m_{01}(B) + m_{01}(A) \times m_{00}(B) + m_{00}(A) \times m_{01}(B)$ $m_{11}(S) = 1 - \left(m_{01}(S) + m_{10}(S) + m_{00}(S)\right)$
XOR $c_0(S) = c_0(A) \times c_0(B) + c_1(A) \times c_1(B)$ $c_0(S) = 1 - c_1(S)$	$m_{00}(S) = m_{00}(A) \times m_{00}(B) + m_{01}(A) \times m_{01}(B) + m_{10}(A) \times m_{10}(B) + m_{11}(A) \times m_{11}(B)$ $m_{01}(S) = m_{00}(A) \times m_{01}(B) + m_{01}(A) \times m_{00}(B) + m_{10}(A) \times m_{11}(B) + m_{11}(A) \times m_{10}(B)$ $m_{11}(S) = 1 - \left(m_{01}(S) + m_{10}(S) + m_{00}(S)\right)$

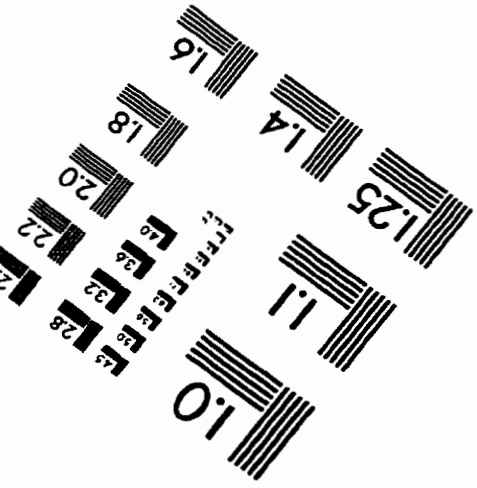
IMAGE EVALUATION
TEST TARGET (QA–3)

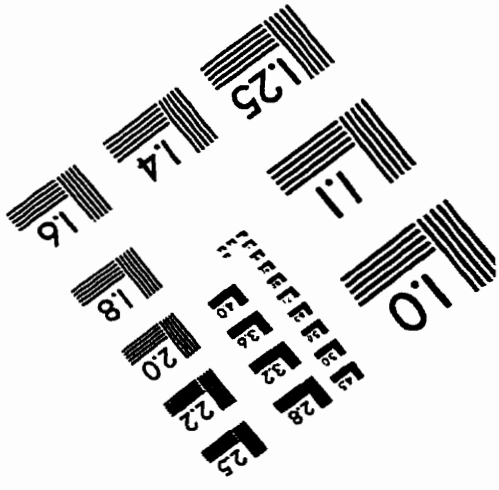












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved